HDRI 3D

CG TIPS AND TRICKS • FOR AND BY CG PROFESSIONALS

ISSUE 9

INSIGHTS & TECHNIQUES FROM PROS:

Integrating LightWave and Vue
Weathering Matte Paintings
Nuke Automation Ideas
Maya 7 Full Body IK

ICE AGE THE MELTDOWN

Issue 9 $10.95 USD $13.50 Canada

© 2006 Twentieth Century Fox Film Corporation

SUBSCRIBE TO

HDRI 3D

MAGAZINE

Lighting the way for digital artists, *HDRI 3D* presents real world insights and emerging trends in digital production from the digital artist's perspective. Concepts and techniques are shared through production case studies and software specific tutorials by professionals in film production, architecture, advertising, medicine, and other fields. Topics are wide-ranging, covering areas such as Workflow, Visual Effects, Compositing, Lighting, Rendering, Modeling, Scripting, Animation, Character Rigging, Texturing, and more.

Go to www.hdri3d.com for more info!

ONLINE SPECIAL OFFER

☐ US $49 ☐ Rest of World $68

Subscribe to *HDRI 3D* Magazine
online before May 31, 2006
to get this great discount!

Six issues a year

4 EASY WAYS TO ORDER

1. In our online store at www.dmgpublishing.com
2. Fax to 1-702-992-0471
3. Phone 1-702-990-8656 or 1-877-303-4629
4. Mail to:

DMG Publishing
2756 N. Green Valley Pkwy #261
Henderson, NV 89014
USA

Name ____________________
Address ____________________
____
City/State ____________________
Zip/Postal Code ____________________
Country ____________________
Telephone ____________________
Email ____________________

PAYMENT

☐ I'm enclosing a check in US Dollars payable to DMG Publishing
☐ I wish to pay by credit/debit card

Card# ____________________ Exp Date __________
Signature ____________________ Date __________

CG TIPS AND TRICKS • FOR AND BY CG PROFESSIONALS

From the Editor
From the Editor
From the Editor

Dariush Derakhshani
Editor-in-Chief

CHANGE IS BLOWIN' IN THE *(DYNAMICALLY SIMULATED)* WIND

A PARANOID RANT

Lots of changes are coming. Run for the hills, grab your rifle, and take lots of soda with you. Autodesk has bought Maya, and everyone is wondering what will be the next logical step for Autodesk to make with Maya. A thought: if Autodesk immediately cuts Maya from its product line, current Maya houses will still be able to sustain themselves and excel for quite some time to come. Maya has a comfortable user base now, and is open-natured enough to allow a plethora of tools to be added to it by that user base for a long time to come. So there is no fear of Maya falling off the face of the planet anytime soon, though you'd never know, anything can happen. I don't think Maya will go away at all even in the longer term, but to stay strong, Maya has to face its weakest asset, rendering. I am extremely keen on seeing what Autodesk is planning (64 bit anytime soon? Pretty please?), and I hope they will work rendering options for Maya. mental ray, Turtle, Maxwell, and now RenderMan for Maya are great inroads; now let's start seeing VRay and Brazil coming into that fold, too, shall we?

And more change: Pixar owns Disney now. Oh, wait, it's the other way around, Disney is acquiring Pixar, and that my friends, makes for an exciting time right now. Just like the saying on the streets: if you get sent to prison, make sure to beat the hell out of someone on your first night as soon as you can, to keep from becoming somebody's bitch. Well, true to form (to make a bad analogy), Pixar nixed Toy Story 3 very soon after news of the Disney acquisition, a decision I wholly admire and agree with. It's perhaps nerve-wracking for those working at Disney animation now (Circle 7); I feel bad for those who have the "up in the air who knows what's going to happen to me" sensation that must be permeating about the ranks–as of this writing at least. By the time this issue goes to print, all may have been resolved! But I have to say, Pixar's decision to stop production on Toy Story 3 until it is developed properly to best serve that wonderful story line makes me feel a whole lot better about Disney and Pixar together. I look forward to some earth-shattering things from Disney animated films now. Good luck to all there.

Hm, well, change is coming to Apple machines. Yes, this is old news, but Intel chips inside Macs make a lot of people salivate at the idea of someday running OSX on their PCs, and gasp, Windows on a Mac machine (you never know!), as well as genuinely faster Macs and Powerbooks. Of all the systems I have seen in my day, I have not seen–with the possible exception of Dell's Precision line–a more well-fitted machine than the new G5 machines. When you have great components that work so well together in one box, you get a great machine. And on top of that, Nvidia is upping the ante for Mac-based graphics workstations with workstation video cards for the G5s. I've recently had the distinct pleasure of purchasing a G5 and beautiful Cinema Display to add to my arsenal at home. I think Apple can improve its 3D performance–based on my limited Mac testing with this past generation hardware–but there is much promise in the near term for sure.

But as much as change is wonderful, I have to briefly mention that my old standby Dell Precision M70 mobile workstation is perhaps the best machine I have ever owned. It's not the fastest, doesn't have the most storage or the best graphics of any system (including desktops) I have used, but it has been the best machine overall. I have to hand it to the M70: the fact that it can run CG apps about as fast as my dual Xeon workstation, as well as play *Counterstrike CS* and *Battlefield2* as well as a good gaming system and I can take it around with me, makes it the best system I can think of. I can't wait to see Dell's next step in their mobile Precision line. A better PC cannot be had.

Change the subject? Sure: Super Bowl! It was nice to see more high-end commercials this year. There was a bit of a recess in huge commercials during the Super Bowl the past few years. It's good to see more CG work coming to life in commercials during the big game. Commercials are a great place to see some of the newest methodologies in the CG world. With tight deadlines and demanding clients, commercial CG work is a great place to cut your teeth and keep your interest level piqued in this business. Good to see a resurge in this market overall lately–let's keep up the steam.

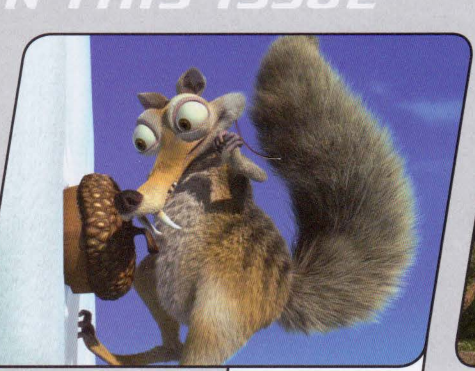
OK, so if I had all the money in the world and all the resources to do what I wanted? After a nice long vacation, I would R&D a way to integrate Shake into Maya, following the trend with Combustion and Max, but taking it much further. As I mentioned in an earlier editorial in Issue 8, it would be way rad if Shake was the interface used to interactively connect render layers together to light a scene. Hell, it need not be Maya, this method would work great with any apps, Fusion and LW, for example, Nuke and XSI. You get my drift.

Even better yet, I would spend time and that fantasy energy figuring out a way to make CG apps as ubiquitous as Photoshop is at this point. I hope the ultimate long-term goal for change for CG applications engineers is to spread CG to the common computing market. This threshold has to be the next market to set sights upon, and I believe this should be the major thought of the times, and how to accomplish this goal.

Until then, the best way to make a wider market for CG is education, and I'm pleased as punch with our progress at HDRI3D magazine. With more tutorials and stories from high-end films like *Madagascar* and *The Lion, the Witch and the Wardrobe* (to name but two of our past issues), it's great to see *Ice Age: The Meltdown* grace our cover with a wonderful image made just for our cover. What an honor, so thanks to the folks at Blue Sky and Fox for making this happen.

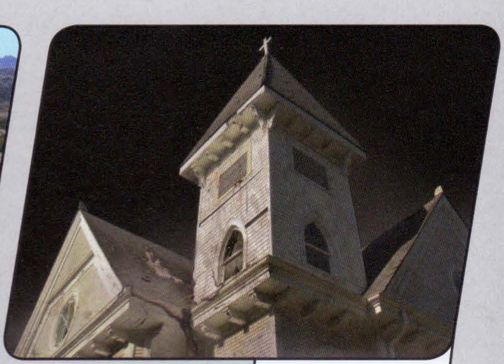
You'll find a lot to read in this issue, so enjoy, and continue to let us know how we're doing and what you'd like to see in future issues. Let us know what matters to you at **www.hdri3d.com**.

IN THIS ISSUE



PAGE 8

PAGE 19



PAGE 22

departments

3 FROM THE EDITOR
BY DARIUSH DERAKHSHANI

6 TD LOVE IN LONDON
BY RAFFAEL DICKREUTER

7 STUFF
BY BRAD CARVEY

12 LIGHTWAVE ESSENTIALS
BY BRAD CARVEY

82 FINAL RENDER
BY ANDREA CARVEY

feature story

8 ICE AGE: THE MELTDOWN
Take a look behind the scenes and see what it takes to create a sequel to the hit CG film *Ice Age*.
BY ERIC KELLER

tutorials

14 USING THE FULL BODY IK IN MAYA 7
MAYA HUMAN ANIMATION FUNDAMENTALS, PART 3
Working with and setting up the new full body IK system in Maya 7, which is based on the Alias® HumanIK® solver used in Alias® MotionBuilder®.
BY PETER RATNER

19 VUE 5 INFINITE
LESSON 2: ADDING A LIGHTWAVE OBJECT TO A VUE ECOSYSTEM
Building on the Vue tutorial in HDRI 3D Issue 8, which showed you how to create cool images with Vue EcoSystems, this issue will show you how to add a LightWave object to the Vue scene.
BY BRAD CARVEY

tutorials

22 WORN OUT LOOKS AWESOME
PART TWO IN A SERIES ON MATTE PAINTING—WEATHERING EFFECTS
This article demonstrates how to create photorealistic effects to weather an existing image of a building.
BY BRIAR LEE MITCHELL

26 TRANSMOTION UTILITIES BASICS
Use an Lscript from the TransMotion Utilities package to transfer a text animation done with After Effects into LightWave 3D.
BY AARON KENT

31 THE TWO FACES OF T-SPLINES
Learn to create detailed, easy to manipulate models using the T-Splines Maya plugin.
BY ERIC ALLEN

36 REAL WORLD PRODUCTION SERIES: LIGHTWAVE AND MOTIONBUILDER
How to send motion capture data from MotionBuilder to LightWave.
BY GREGORY "T.REX" GLEZAKOS

44 FX MAGIC WITH MAYA AND PARTICLEILLUSION
A handy MEL script makes it easy to incorporate particleIllusion's effects with Maya animations.
BY ERIC KELLER

50 MOTION CAPTURE AS A TOOL
Now that motion capture is much more essential in the animation workflow for reality based games, it's our job as animators to push the boundaries beyond what mocap inherently is by using it as a workflow tool.
BY RICHARD LICO

54 GRAB LIGHTWAVE BY THE NULLS!
AN IN-DEPTH LOOK AT LIGHTWAVE'S NULL OBJECTS
In this article we take a look at why Null objects are so useful and explore some of the many uses for them in a LightWave production. Discover how you can start using Null objects in everything you do to improve your workflow.
BY WILLIAM "PROTON" VAUGHAN

PAGE 50

PAGE 54

contents

tutorials

60 MEL SPLASHFLOCK

A few MEL scripts and a model generate an effect called the "SplashFlock." This article goes through the MEL scripts, attribute by attribute, to fully explain the process. Content geared toward novice users.

BY THATCHER BELL

64 HAULIN' WITH MULE

When you are working on a project and are in full production, one of the many delays can be the rendering and test rendering of scenes. What I want to do is take you through the weekly process I go through on my animation projects with MULE, and show you how I saved up to 30 percent of production time using this program.

BY DALE L. CAMPBELL, JR.

68 RIGID BODIES AND SPRING NETWORKS 101

The principles behind a dynamics system and first steps in applying them.

BY RAFFAELE FRAGAPANE

72 AUTOMATING COMPOSITING AND RENDERING TASKS WITH NUKE

This article looks at creating and designing custom Nuke nodes with unique UI's. Essentially these nodes are nested flowcharts within a so-called gizmo. These gizmos can then be loaded into any script, or even included within the Nuke menu as a plugin. Gizmos are used for simplifying repetitive or complex compositing tasks.

BY BOAZ LIVNY

review

78 REVIEW: IBM INTELLISTATION M PRO

A review of the IBM IntelliStation M Pro, one of the quietest and coolest systems around.

BY CHARLES EDGIN

Resource files and selected screen shots for this issue are posted at www.hdri3d.com/resources

HDRI 3D

Issue 9

EDITORIAL DIRECTOR	
Charles Edgin	charles@hdri3d.com
EDITOR-IN-CHIEF	
Dariush Derakhshani	dariush@hdri3d.com
EDITORS	
Brad Carvey	bradcarvey@hdri3d.com
Raffael Dickreuter	raffael@hdri3d.com
MANAGING EDITOR	
Alice Edgin	alice@hdri3d.com
TECHNICAL EDITOR	
Victor J. Garza	victor@hdri3d.com
ART DIRECTOR	
David S. Wilkinson	david@sabredesign.net
COPY EDITOR	
Cheri Madison	cheri@hdri3d.com
LAYOUT/DESIGN	
Sabre Design	artists@sabredesign.net
CONTRIBUTORS	
Eric Allen	ericmatthewallen@hotmail.com
Thatcher Bell	zolividor@hotmail.com
Dale L. Campbell, Jr.	dale@positiveproductionsplus.com
Andrea Carvey	andreacarvey@hotmail.com
Brad Carvey	bradcarvey@hdri3d.com
Dariush Derakhshani	dariush@hdri3d.com
Raffael Dickreuter	raffael@hdri3d.com
Charles Edgin	charles@hdri3d.com
Raffaele Fragapane	jaco@thejaco.com
Gregory "T.Rex" Glezakos	gregory@sub-pixel.com
Eric Keller	kellerrific@yahoo.com
Aaron Kent	kindofshiftless@yahoo.com
Richard Lico	richard.lico@gmail.com
Boaz Livny	info@visionanimations.com
Briar Lee Mitchell	briarLee@starmountainstudios.com
Peter Ratner	ratnerpj@jmu.edu
William "Proton" Vaughan	proton@spinquad.com

HDRI 3D MAGAZINE ADVISORY BOARD
Dan Ablan • Timothy Albee • Akiko Ashley
Jack "Deuce" Bennett II • Stephen M. Burns
Kevin Cahill • Dwayne J. Ferguson
Todd Grimes • Ed Harriss • Varto Keshishian
Policarpo • William Vaughan

ADVERTISING SALES	1-702-990-8656 advdmg@dmgpublishing.com
SUBSCRIPTIONS & BACK ISSUES	
Online Store	www.dmgpublishing.com
Email	mag@hdri3d.com
Phone	1-702-990-8656 or 1-877-303-4629
Fax	1-702-992-0471
CHANGE OF ADDRESS	newaddress@hdri3d.com

HDRI 3D MAGAZINE
(ISSN 1551-689X)
is published bimonthly by
DMG Publishing
2756 North Green Valley Pkwy #261
Henderson NV 89014 USA

© 2006 by DMG Publishing. All Rights Reserved. No part of this publication can be reproduced or transmitted in any form or by any means, electronic or mechanical including photocopying, recording, or information storage and retrieval system without permission in writing from the publisher.

TD Love in London

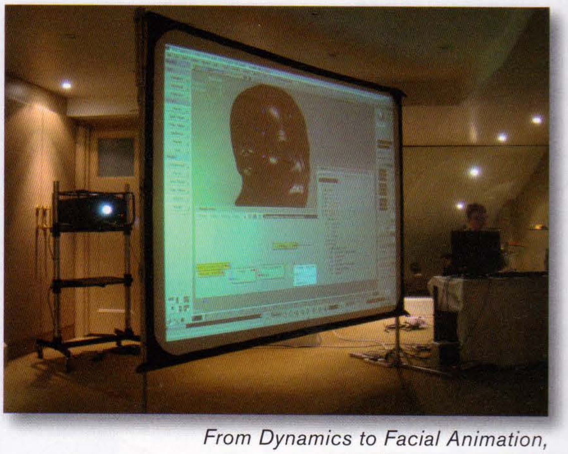
Softimage's TD Love tour stopped in London.

Raffael Dickreuter
Editor

Nowadays more and more public relations is moving to the web. You can download the latest software update, read the new features lists; you probably won't get an upgrade package anymore. At these times it's still important to stay in touch with your industry peers, as well as customers, on a regular basis face to face. A great place to do that is by visiting events, user group meetings, and industry parties.

Softimage became very active in that area in the past year. After announcing XSI 5.0 as well as Face Robot, they started to tour the world with the "3D Love Tour" to show 3D artists everywhere the exciting new tools. Events were held in New York, Los Angeles, San Francisco, Vancouver, Montreal, Paris, London, and many more.

But that wasn't all. Just when the 3D Love Tour was over, another one was started: "TD Love," which is basically a one day technical workshop focusing on XSI as well as third-party tools for the software.



From Dynamics to Facial Animation, many topics were covered.

On January 10, the TD Love event in London took place. Softimage had already toured the world in the previous month visiting many CG metro poles and holding this one-day event.

Softimage also offered a big buffet of sandwiches, snacks, and drinks.

Hands-on XSI training in London, UK.

The first session featured Kim Alids, who talked in depth about rigid bodies. The problem to solve was a newspaper spot, where letters and phrases would fall off the paper as easily as sand. The text is broken apart into single letters using scripting; the text is still parented under a null to keep everything together as long as it should be. A random but controlled selection with a script helps to feather the lower part of the text, which is first to fall off.

To keep the text together, all the letters are first passive rigid bodies, then the type is animated so they suddenly become active rigid bodies and start to fall. Another tip was to avoid parenting rigid bodies as it might influence and screw up the animation and give unpredictable results.

Kim also mentioned to use boxes rather than grids for obstacles as grids seem not always to work and objects fall through. Also, more dense subdivision helps to get better results.

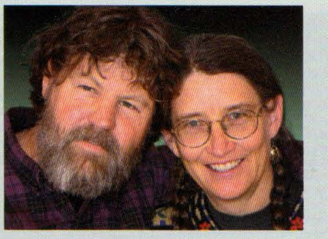
James Rogers of Softimage demoed Softimage|FaceRobot, which showed interface improvements since last SIGGRAPH. It's tightly integrated into XSI's interface and shows options on the right menu side, as well as a progress bar for your picking session. As shown earlier, you can delete points you picked and you will be prompted to select the needed key points again, so Face Robot can generate the rig. He also showed some clips of a mocap session with the actor and then the face of Rock Falcon with the animation applied to it. It showed how well the animation and performance transfer, including many subtleties.

Alan Jones talked about transfer from ZBrush to XSI and how you can get the displacements maps generated in ZBrush into your scene. He also talked about a dynamic setup for facial animation sliders. It combines sliders and f-curves and can be easily modified on the fly as it is based on scripted operators. The approach of having sliders and f-curves at hand makes it much more flexible to blend different shapes.

Harry Bardak's presentation focused on rendering and showed shader creation using the plugin manager, how you can easily generate an SPDL for your shader with it. He showed that you can get final gathering like results using ambient occlusion by plugging it into the volume node of the material.

These kinds of events are always a great way to learn something new and to meet up with peers from the industry and stay up to date with what's happening. People move from project to project and from company to company. There is always stuff happening, and new technologies are being presented. Going to events such as TD Love shows why computer graphics is such an exciting field to work in.

STUFF



By Brad Carvey
Electrical Engineer, 3D Animator
New Mexico

COLOR CONTROL

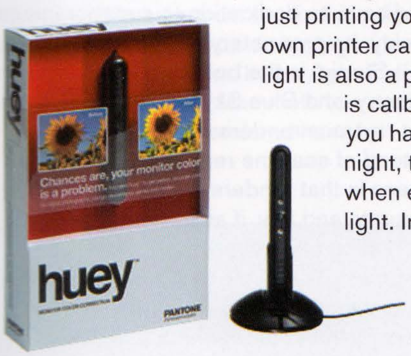
Did you ever wonder what the Custom button was for in Photoshop's Color Picker? If you have never selected "Custom Colors," then you are probably not aware of Pantone colors. Pantone colors are standard colors defined by a company called Pantone. For example, PANTONE 179 C is a pinkish red color. You can buy sets of cards that have Pantone colors on them with their standard names. So if you want to match a color of a real shirt to a color in the computer, you can search for the card that is the same color as the shirt. Once you know the Pantone color name, then you can select the same color, using Photoshop's color picker's custom colors. If you hold the shirt up next to the monitor, the two colors will probably look like two different colors. This is because your monitor is not calibrated.

Most of us adjust our monitors until they look good. There are times that I need to adjust the color of an image a few times before the printer's colors look even close to what they should be. Print professionals use very expensive equipment to calibrate their monitors so their printouts look like the monitor and have accurate colors.

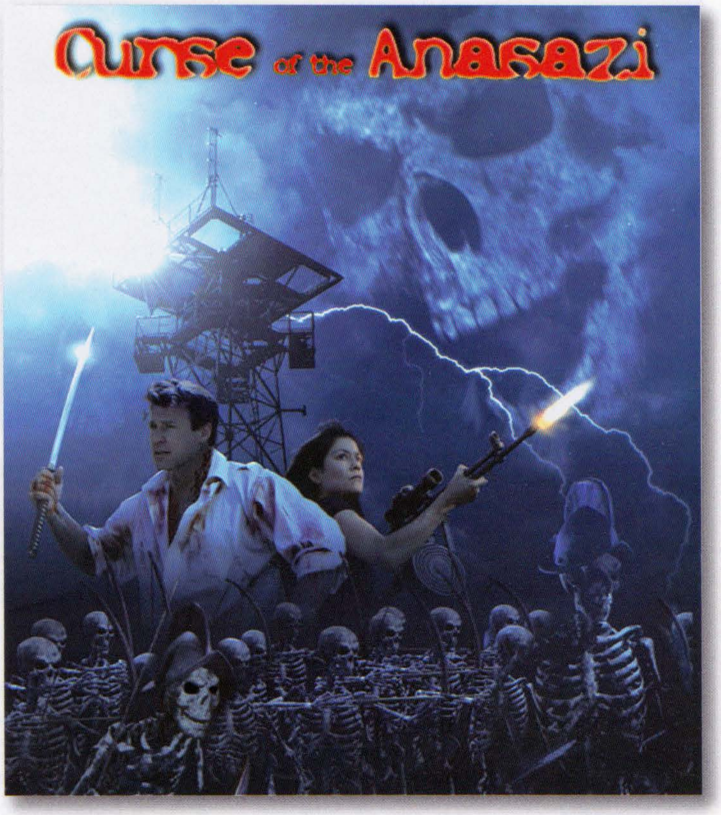
CRT monitors have had Red, Green, and Blue dots grouped together to create colors. The reason RGB colors are used is because in the early days of Television, the Red, Green, and Blue phosphors were the easiest and brightest available. For the human eye to see white, the light would need to be 11% Blue, 30% Red, and 59% Green. Starting to sound complicated.

With print they don't use RGB; instead they use CMYK (Cyan-Magenta-Yellow-Black). CMYK is known as 4-color printing. My Canon Pixma iP5000 has CMYK plus a special photo black. Converting RGB colors to CMYK or just printing your RGB images with your own printer can be difficult. Ambient light is also a problem. If your monitor is calibrated for daylight and you have florescent lights on at night, then the colors will be off when exposed to the florescent light. In the past most of us probably did not bother trying to calibrate the monitor.

I recently found a device that solves all of the color correction and ambient light compensation problems. The $89 color controller, called Pantone huey, is a must have for anyone serious about computer graphics. I don't have one yet, but will be buying one right away. Calibrating your LCD or CRT monitor has been primarily guesswork for a lot of graphics artists; with Pantone huey, the guesswork is a thing of the past.



Pantone huey is a USB powered device called a colorimeter that measures the color from your LCD or CRT monitor. It includes software that adjusts the color of your monitor. It also checks the ambient light periodically and adjusts the color. With Pantone huey your printouts should look the same as what you see on your monitor. When I get one, I will probably write a review.



CURSE OF THE ANASAZI

Joe Conti was one of the first Video Toaster and LightWave users. Back in the late 80's, I was working on what was eventually called the Video Toaster. Joe was visiting his Dad in Albuquerque, after graduating from Film School. I showed him the first Toaster proof of concept prototype. Joe moved to L.A. and immediately started using the Toaster to create visual effects for feature films. Go to www.contifilms.com to see some of the work that he has done over the years. A few years ago Joe directed his first movie. It was a SciFi Channel original called Bugs. Joe co-wrote, directed, and did all of the visual effects for the Bug movie. I believe he did over 300 compositing and animated 3D bug shots for his movie.

Now Joe is creating his second movie, called Curse of the Anasazi. Joe wrote the story, directed the movie, and is currently co-editing it. Joe even paid for everything with his own money. I have seen some scenes with people fighting sword-wielding 3D Skeletons and they look great. Joe will probably do another 300+ shots for this movie as well. With inexpensive HD equipment and a computer, talented creative people like Joe can now make a high-quality movie with minimal help.

To minimize the cost of the movie, Joe hired about six key people who help supervise a crew of trainees. The strategy worked and Joe was very happy with how smoothly the very tight 15-day shooting schedule went.

Paul Montgomery once said that in the future your favorite movies will be made by yourself or someone you know. For me the future has arrived.

ICE AGE THE MELTDOWN
TM & ©2006 TWENTIETH CENTURY FOX
ALL RIGHTS RESERVED.

ICE AGE: THE MELTDOWN

By Eric Keller
Freelance Maya Animator
West Hollywood, California

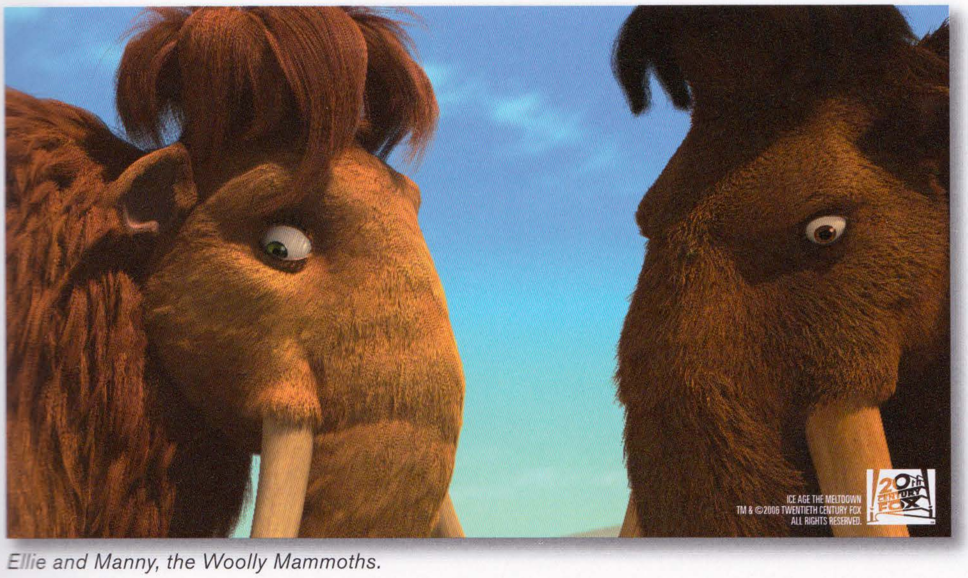
While creating a sequel to the hit feature film *Ice Age*, the team of CG artists at Blue Sky Studios in New York immediately faced two competing challenges: On the one hand, a sequel to a spectacular film demands to be bigger and even more spectacular; on the other hand, every part of the sequel–from the environment to the mannerisms of the returning characters–must be consistent with the original. In addition to these challenges, the technology used to create CG films has changed a great deal in the four years since the making of the original *Ice Age*.

HDRI 3D talked with some of the team at Blue Sky about the making of *Ice Age: The Meltdown* and how they topped the original film while maintaining familiarity with the world they created back in 2002. In a conversation late on a Friday afternoon and during the final crunch to get the film finished, the artists were nevertheless enthusiastic about sharing their experiences. We spoke with Modeling Co-Head Shaun Cusick, Fur & Feathers Supervisor Eric Mauer, Materials Supervisor Mike Eringis, Effects Supervisor Rob Caveleri, Supervising Animator Galen Chu, CG Supervisor Mike Travers, and Senior Lighting Lead Andrew Beddini. Each of them had a lot to say about the part they played in creating *Ice Age: The Meltdown*.

VISUAL PERFECTION FROM THE VERY BEGINNING

Blue Sky Studios was founded in 1987 and became part of 20th Century Fox 10 years later. Since day one of Blue Sky's existence, their guiding principle was a dedication to creating the highest quality, most photorealistic images possible with computer graphics technology. This is true even when the project is a stylized animated odyssey, as the *Ice Age* movies are. Blue Sky is a closely-knit group of artists. Virtually every aspect of the animation is done in-house, from storyboarding to animating, rendering, and editing. The studio recruits most of its staff locally in and around New York, drawing on the expertise and talent that has developed in the surrounding commercial and effects studios. There are about 350 staff at Blue Sky, 250 of whom are traditional and computer artists working on the film.

Along with the talent of the artists, what sets Blue Sky apart from other CG film studios is its dedication to superior image quality. Its proprietary rendering software, CGI Studio, is the best raytracer in the industry, and Blue Sky is the only studio that raytrace renders entire feature films instead of scanline rendering. What this means is that renders are more physically accurate and few, if any, cheats are used in creating reflections, refractions, and shadows. Raytracing is a process where a CG camera shoots out rays into a virtual scene. When the rays hit a reflective or refractive surface, they split into additional rays. When the rays bounce off a reflective surface onto a non-reflective surface, this second surface appears as a reflection on the first. There are many raytracing renders available to digital artists, but few boast the power and flexibility of Blue Sky's CGI Studio, which has its own powerful script-based application programming interface (API) that allows the artists to further extend the software's capabilities and to adapt it to the needs of their scene with scripts and custom plugins written in-house.



Ellie and Manny, the Woolly Mammoths.

IMPROVEMENTS IN TECHNOLOGY MEAN FATTER FUR

Since the first *Ice Age* film, CGI Studio has gone through four major version upgrades—and it's not the only Blue Sky technology to have been upgraded. Every department has expanded and improved to create the increased level of detail that sequel demands. When the original *Ice Age* was created, Blue Sky was just developing its CG feature film pipeline. Over the past four years, the studio has created, expanded. and refined new tools and techniques used on *Ice Age: The Meltdown*. For example, Manager of Technical Software Maurice van Swaaij completely rewrote the fur technology so that each hair is drawn in the software, voxelized, and rendered. This is a huge difference from the first *Ice Age*, which relied on sprite cards arranged in clumps to create the fur. In *Ice Age: The Meltdown*, Manny the Mammoth has some 2 million individually rendered hairs on his back. Procedural algorithms allowed the hair to dynamically create secondary animation when Manny moves around. However, Manny's new hairdo was in danger of breaking the consistency of his look from the first film to the second. In order to avoid the visual disconnect, the team working under Fur & Feathers Supervisor Eric Maurer described the fur for all the characters using vector and color fields generated by a suite of custom nodes in Maya. These nodes were created based on the fur models generated in the original film, and the information was then piped into the new fur technology.

NEW TOOLS FOR MODELING

Blue Sky continues to incorporate Alias's Maya into its modeling and animation workflow. The artists have expanded Maya's capabilities through the use of numerous MEL scripts and plugins developed in-house by the Research and Development team. The first *Ice Age* used Maya versions 1 through 2.5. *Ice Age: The Meltdown* is finishing production with Maya version 6.5. Needless to say, much has changed over the past four years in Maya. The most significant change has been a movement away from characters built with NURBS patch models and towards characters created with a single subdivision surface mesh. This means that all the returning *Ice Age* characters have to be completely rebuilt while remaining true to the original models. In addition, the artists designed and modeled new central and supporting characters. Blue Sky takes advantage of the many artists on its staff with experience in traditional animation and stop motion. In designing the characters, they built clay maquettes and scanned them into the computer to serve as modeling templates. *Ice Age: The Meltdown* required the construction of three times as many clay maquettes as the first film. These maquettes were also laser scanned, instead of the older method of digitizing the model with a stylus connected to an armature.

RIGGING FOR COMPLETE CONTROL

All the new models also required new rigs. The characters each used a single all-purpose rig throughout the film, with the exception of Manny the Mammoth, who occasionally required a second IK rig setup for his trunk in some scenes. These rigs required an enormous number of controls to allow for the maximum amount of flexibility when animating. Modeling head Shaun Cusick noted that the difference in file size between the model by itself and the model with all of its rigging and animation controls was often 15 to 20 times. Part of the reason for this is that the technical directors at Blue Sky prefer to use articulated facial rigs created with bones that have a very fine level of control instead of the more common technique of using blend shape targets to interpolate between facial poses. They make this choice because of the increased flexibility an articulated rig gives to an animator. Blendshape targets can take a long time to create and, often, if the supervisor or director wants to make a change quickly, the blendshape targets may need to be altered or rebuilt, which can be a significant drag on resources. The talented technical directors at Blue Sky are able to create a facial rig with enough controls and power so that creative changes can be absorbed easily.

CONTINUITY IN CHARACTER

Animation Supervisor Galen Chu ensured stylistic consistency with the original film by having his animators use a custom-built pose toolbox. Key character poses were saved directly from the original files from the first film and stored in this digital toolbox.

Fast Tony the Armadillo.

ICE AGE THE MELTDOWN
TM & ©2006 TWENTIETH CENTURY FOX
ALL RIGHTS RESERVED.
20th CENTURY FOX

ICE AGE THE MELTDOWN
TM & ©2006 TWENTIETH CENTURY FOX
ALL RIGHTS RESERVED.

Scrat the Saber-Toothed Squirrel.

The animators then used these poses as a template. The templates were the starting point for animation on the sequel, thus ensuring that the new and improved Sid the Sloth still moves like the Sid the Sloth we grew to love in the original film. None of his personality was lost in the upgrade to the new technology (no doubt to the dismay of his companions, Manny the Mammoth and Diego the Saber-Toothed Tiger). The rigs created for the animators also had a great deal of squash and stretch compensation built in. For example, if they stretch out Sid's neck, the radius of the neck geometry automatically contracts because of the controls built into the rig. Likewise, the radius of the neck expands if his head is squashed down into his torso.

Chu said the animation style at Blue Sky is inspired by a number of influences, including the cartoons of Chuck Jones, Tex Avery, and classic Disney, as well as the CG films of Pixar and DreamWorks PDI. However, for the most part, Blue Sky is interested in doing its own thing. The artists animate based on what feels right to them without trying to pay homage to any particular animator or studio. The demands of the scene are the biggest factors in determining how a character will be animated. This means, for instance, that Sid the Sloth will be more cartoonish in some scenes than others, depending on the action in the scene and the performance of the voice actor, John Leguizamo.

SKY LIGHTING

According to Senior Lighting Lead Andrew Beddini, the biggest change from the first movie to the second is the lighting. Blue Sky's proprietary lighting system is amazingly efficient and powerful. For *Ice Age: The Meltdown*, much of the ambient shadowing techniques that add realism to outdoor sequences have been built directly into the rendering system. This allows them to light an outdoor scene with just three lights, while the same look might take other rendering technologies thousands of lights and much more complicated setups. The lights are also based on camera position instead of on world space coordinates. This means that a lighting technical director can easily set up a light at an angle of 30 degrees from the camera's position and 100 units back in precisely those terms without having to try and figure out what the same position would be in world space.

THE MELTDOWN

Without giving away too much of the plot, it is safe to assume with meltdown in the title that there will be a great deal of water in the film. Indeed, simulating water was the biggest effect the artists had to deal with. Beddini said that the large number of underwater scenes posed some of the biggest lighting challenges. But Blue Sky took a novel approach to solving the many water problems. The solution started with the artists' desire to create very realistic water directly in the render. Underwater ambient lighting was created using 100% physically based simulations. In fact, rather than animate characters over a blue background with typical visual clues to indicate an underwater scene, the artists actually created an accurate physical simulation of a body of water in the computer and submerged the characters, cameras, and lights to create extremely realistic underwater scenes. This means that the bubbles and the underside of the surface behave based on accurate simulations, and that they reflect and refract light exactly as they would in reality without the artists' using any cheats whatsoever. A bubble in Blue Sky's underwater scene is not a spherical piece of geometry with some soft-body dynamics applied; rather, it is an actual pocket of space in a simulated body of CG water that behaves accordingly.

To create the look of wetness, the artists took advantage of CGI Studio's procedural texturing methods. In fact, every surface texture in the film is created procedurally. There are no Photoshop artists painting texture maps for any of the models. This method was developed during Blue Sky's work on *Robots* and has been further expanded in *Ice Age: The Meltdown*. It is a credit to the power of CGI Studio that most of the default settings on their materials are already very realistic. This means that texture artists spend less time trying to create a realistic wet or dry look for a surface and more time on surface patterns and colors to match the vision of the art directors. Because of this, designing the look of the water was actually very easy; realistic results were almost immediate.

SPLASH CONTROL

Animating the water was another story altogether. CG films can sometimes turn into one large special effect, especially

Diego the Saber-Toothed Tiger holding Sid the Sloth by the neck.

ICE AGE THE MELTDOWN
TM & ©2006 TWENTIETH CENTURY FOX
ALL RIGHTS RESERVED.

ICE AGE THE MELTDOWN
TM & ©2006 TWENTIETH CENTURY FOX
ALL RIGHTS RESERVED.
20th CENTURY FOX

Crash and Eddie, the Possums.

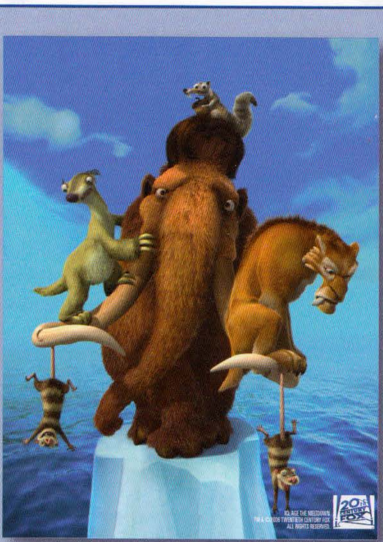
when you have a dynamic and interactive environment such as the one in *Ice Age: The Meltdown*. There are many different types of water that required animation and, often, the characters needed to be interacting with the water. Effects Supervisor Rob Caveleri took the approach of treating the water as a character. His goal was to create a system in which the water could be directable rather than relying on the unpredictability of fluid simulations. Blue Sky used fluid simulations created with Next Limit's RealFlow as a starting point, along with a great deal of reference footage. From there, they used what Caveleri describes as "a grab bag of tools and techniques" to come up with a unique and stylized approach to animating the water. Precise control over splashing water was achieved through the use of a key frameable patch animation system. The effects animators used a series of controls to move the water around like a character. In scenes where characters interact with water, the fluid animation would get approval first and then a lower resolution version would be passed on to the character animators. The characters could then be animated moving about, and from there the water could be further refined. The biggest problem with CG water is that it often looks too perfect; this is what leads viewers to say that CG water looks less realistic. The artists at Blue Sky developed ways to create realistic frothiness based on the cavitations of the water surface as characters moved through it.

A LARGE SMALL TEAM

A staff of 250 artists seems very big until you consider that everything that appears on the screen in an all-CG movie must be built, textured, lit, animated, and rendered. Nothing comes for free in this situation. On top of this, audiences' expectations for CG films rise each year as the major studios continue to produce one breathtaking breakthrough animated film after another. In the thick of making a CG film, a crowd of 250 people suddenly becomes a closely knit group. Often the demands of the project require that various departments trade personnel, and everyone ends up pitching in on various tasks. At the time of our conversation, Blue Sky was wrapping up the effects work and the artists were "knee deep" in lighting and rendering the film. Many of the departments who had borrowed lighting technical directors at an earlier stage were now lending a hand with the enormous task of lighting. It helps that most of the artists at Blue Sky are experienced in many aspects of CG. Although there is specialization, the artists interact with each department and are not completely isolated in an assembly line.

With the all-CG film now established as its own genre, it is becoming increasingly common for sequels to be created for the most successful films. CG technology is constantly expanding and improving in an attempt to keep up with the higher demands of appreciative audiences. Since 1987, Blue Sky Studios' dedication to pushing the envelope in realistic CG rendering, coupled with its talented artists who are adept at making stylized feature films, ensures that they remain at the forefront of CG filmmaking. The enthusiasm the artists displayed while talking about their work on *Ice Age: The Meltdown* is a testament to their ability to face the double challenges of creating a bigger, more fantastic sequel while still keeping the spirit of the characters and the world that made the original so memorable.

Eric Keller *is a freelance animator living and working in West Hollywood, California. He has an amazingly patient wife and two great dogs. He has recently contributed two chapters to the upcoming* Mastering Maya 7 *book published by Wiley.*



THE MELTDOWN

In *Ice Age: The Meltdown*, our sub-zero heroes from the worldwide blockbuster CGI film *Ice Age* are back–Manny the Woolly Mammoth, Sid the Sloth, Diego the Saber-Toothed Tiger, and the hapless saber-toothed squirrel known as Scrat. In the new film, from the Academy Award winning creators of *Ice Age* and *Robots*, the Ice Age is coming to an end, and the animals are delighting in the melting paradise that is their new world.

Ray Romano, John Leguizamo, and Denis Leary return to voice our three heroes: Manny, Sid, and Diego. New cast members include: Queen Latifah, Seann William Scott, Jay Leno, Will Arnett, and Josh Peck.

Our trio is still together and enjoying the perks of their now melting world. Manny is ready to start a family, but nobody's seen another mammoth for a long time. Manny thinks he may be the last one. That is, until he miraculously finds Ellie (voiced by Queen Latifah), the only female mammoth left in the world. The only problem is: they can't stand each other.

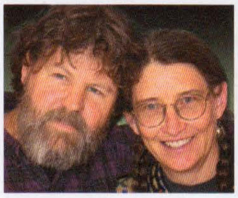
Ellie comes with some excess baggage in the form of two possums... Crash and Eddie (voiced by Seann William Scott and Josh Peck), a couple of daredevil pranksters. These are two cocky, loud-mouthed troublemakers.

Manny, Sid, and Diego quickly learn that the warming climate has one major drawback: a huge glacial dam is about to break, threatening the entire valley. The only chance of survival lies at the other end of the valley. So our three heroes, along with Ellie, Crash, and Eddie, embark on a mission across an ever-changing, increasingly dangerous landscape towards their salvation.

The film opens March 31, 2006.

LIGHTWAVE ESSENTIALS

NULL CONTROL



By Brad Carvey
Electrical Engineer, 3D Animator
New Mexico

The key to animating non-organic mechanical objects is to set up an appropriate hierarchy of object parts and nulls. Using different nulls to control a variety of rotations, movements, and deformations allows the artist to modify each individually, without affecting the others.

The Unicylce for our flip-book animation.

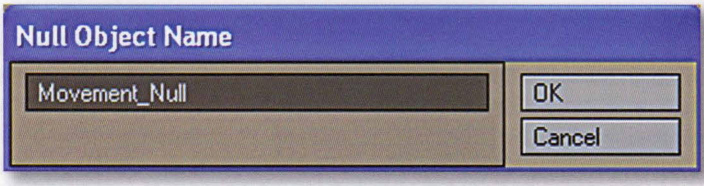
LOAD THE UNICYCLE AND ADD THE NULLS

1. Download unicycle.lwo from the Resources page at *http://www.hdri3d.com/resources/index.htm.*

2. Load the object into LightWave.

3. Add a Null (Ctrl + N) and name it Movement_Null (**figure 1**).

4. Add four more Nulls, naming them Jump_Null, Steering_Null, Lean_Null, and Squash_Null.



Null Object Name

Movement_Null

Figure 1.

SET UP THE OBJECT AND NULL HIERARCHY

1. Open the Scene Editor (Ctrl_F1).

2. Select Jump_Null, drag it over the Movement_Null, and release the mouse button.

3. Select Steering_Null, drag it over the Jump_Null, and release the mouse button.

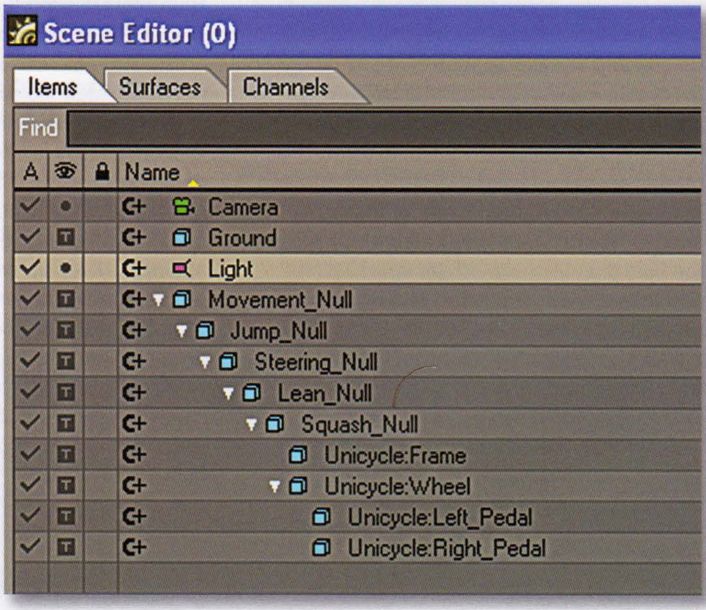
4. Select Lean_Null, drag it over the Steering_Null, and release the mouse button.

5. Select Squash_Null, drag it over the Lean_Null, and release the mouse button.

6. Select unicycle:Frame, drag it over the Squash_Null, and release the mouse button.

7. Select unicycle:Wheel, drag it over the Squash_Null, and release the mouse button.

8. Select unicycle:Left_Pedal, drag it over the unicycle:Wheel, and release the mouse button. Do the same for the Right_Pedal (**figure 2**).



Scene Editor (0)

- Camera
- Ground
- Light
- Movement_Null
  - Jump_Null
    - Steering_Null
      - Lean_Null
        - Squash_Null
          - Unicycle:Frame
          - Unicycle:Wheel
            - Unicycle:Left_Pedal
            - Unicycle:Right_Pedal

Figure 2.

ANIMATION WITH THE NULLS

1. Select Movement_Null, move the unicycle 4 meters along the X-axis, and create a keyframe at frame 60.

2. Select Unicycle: Wheel, move to the last frame of your animation, and rotate the Pitch to 720 degrees.

3. Select the Lean_Null, move to the frame 0, and rotate the Bank to 18 degrees. Rotate the Bank on every frame where the pedal is down (15, 30, 45, and 60). The unicycle will rock back and forth based on the pedaling action.

4. Select the Steering_Null and rotate the Header as the pedals turn. As the pedal goes down, the unicycle will tend to steer in that direction.

Excellent NAB Specials!!

Avid
Liquid 7
Xpress
Bauhaus
Mirage
eyeon
Fusion5
G-Technology
Drive Arrays
Medéa
Drive Arrays
Training
DVD's, Books, 1on1

NewTek
Lighwave 3D
TriCaster
VT[4]
Serious
Magic
DV Rack
Ultra2
Visual Communicator
Turn-Key
Systems
Laptops
Upgrades

11300 Hartland Street North Hollywood, CA 91605 818-760-4445 www.vgn.com

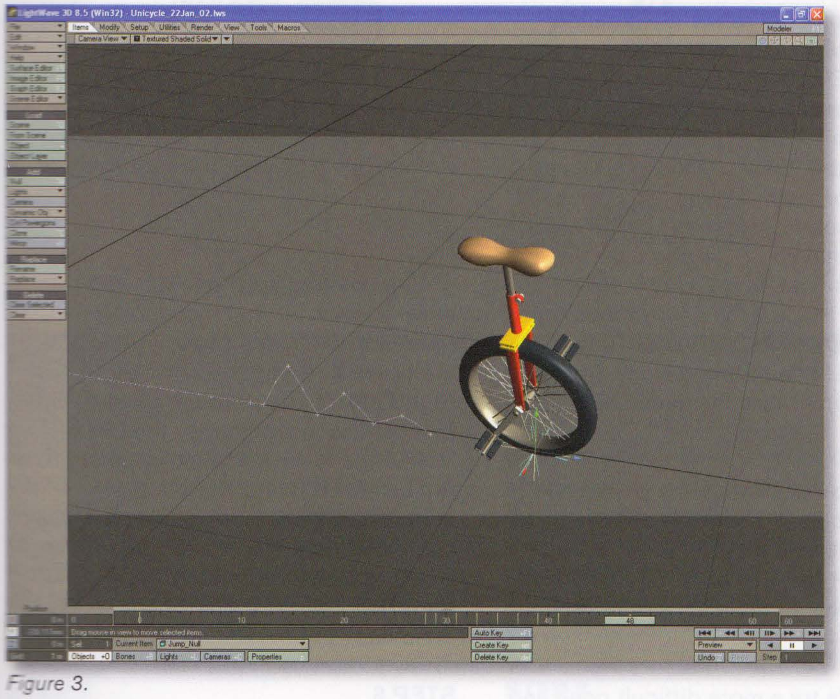


Figure 3.

At this point the unicycle is rolling along the ground. It is leaning from side to side and rotating a little as it moves along. The advantage with the nulls is that we can make small changes to one of the nulls without affecting the movement or rotation of the other nulls.

USING THE JUMP NULL

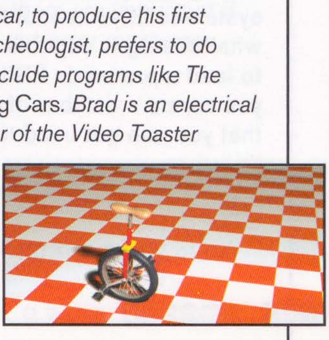
1. Select the Jump_Null, move to frame 28, and create a keyframe. Move to frame 28, and create another keyframe. Make keyframes for frames 33, 37, and 41.

2. With the Jump_Null selected, move to frame 31, move the Jump_Null up in the Y direction, and create a keyframe. Repeat at frame 35 and frame 39, with the Y distance about half for each frame (**figure 3**).

The Unicycle: Frame can be rotated in the pitch to give the feel of rocking the frame forward and back. It's also possible to use the Squash_Null to compress the unicycle as it starts to jump and expand it at the height of its jump. The pedals can also be rotated. Additional nulls can be added to the hierarchy to control additional movements.

I moved the camera and added some lighting, then rendered a 60 frame animation. If you flip the pages of the magazine you will be able to see the animation that I created.

Brad and Andrea Carvey *have been doing computer animations for a long time. In 1969, Brad used an analog computer, which was the size of a car, to produce his first computer animation. Andrea, an archeologist, prefers to do scientific animations. Her credits include programs like The Discovery Channel's* Understanding Cars. *Brad is an electrical engineer and an Emmy Award-winning member of the Video Toaster development team. He prefers to do feature film work. His credits include films like* Men in Black, Stuart Little, Black Hawk Down, Kate & Leopold, *and* Master of Disguise.



Using the Full Body IK in Maya 7

By Peter Ratner
Professor, Artist
Penn Laird, Virginia

Part 3 in the series of Maya Human Animation Fundamentals. Parts 1 & 2 can be found in *HDRI 3D* Issues 7 & 8.

Even though the plan for this issue's article was to use the Graph Editor in Maya to edit Trax clips, I believe it's more important to talk about using the new Maya 7 full body IK system. For those of you who have been following the previous issues' instructions on rigging and have upgraded to Maya 7, it is no longer necessary to go through so many steps to build an IK system. The full body IK system in Maya 7 is based on the Alias® HumanIK® solver used in Alias® MotionBuilder®. Once you start working with this rig, you should find it superior to any previously constructed system. It's an impressive addition to Maya that by itself is worth the upgrade.

PART 1: WORKING WITH THE FBIK RIG

Before creating your own skeleton and applying the full body IK (FBIK) system to it, you should see for yourself what this rig is capable of. We're going to load a pre-made FBIK rig and have you create a number of poses with it so that you can get used to working with the system.

STEP 1

Go to ***Window > General Editor > Visor*** and click the FBIK Examples tab. Right-click on the bipedExample.ma file and select Import Maya File bipedExample.ma.

STEP 2

You should now see a fully rigged skeleton that uses both inverse and forward kinematics. The round spheres are called Effectors. These can be translated to move the skeleton using full body IK. They can also be rotated to turn parts of the skeleton at the joints, thus giving you forward kinematics control. The effectors can also be anchored in place (pinned) to prevent unwanted movements or rotations during walk cycles and other circumstances. Let's start with an easy pose (**figure 1**). Move the Left and Right hand effectors (LeftHandEff and RightHandEff) above the head joint. To rotate the shoulder joint up, select the LeftArmEff and RightArmEff effectors. They are difficult to see because of their small size, but you can find them in the Outliner in the humanIK1 folder. Once you locate them, scale them larger so that they are easier to select. Scale any of the other effectors that you think might be difficult to find or select. In the Channel Box, select the Scale X, Y, Z attributes, right-click, and pick Key Selected. Rotate or move the LeftArmEff and Right ArmEff effectors up to turn the shoulders.

STEP 3

When you get both arms positioned correctly, select any of the effectors and press Control+f or go to ***Animation > Animate > Set Full Body IK Keys***. This will keyframe all the effectors.

STEP 4

Now let's try a more difficult pose and start using the pinning feature to help us retain the position of specific limbs. Refer to **Figure 2** for the next several steps. On the timeline find the frame where you keyframed all the effectors. Right-click on it and select ***Delete FBIK keys > Delete All FBIK Keys***. When you are animating, you can use this option to delete keys anywhere on the timeline. To place your rig back to its neutral T-stance, select a joint and go to ***Animation > Skeleton > Full Body IK > Go to Stance Pose***.

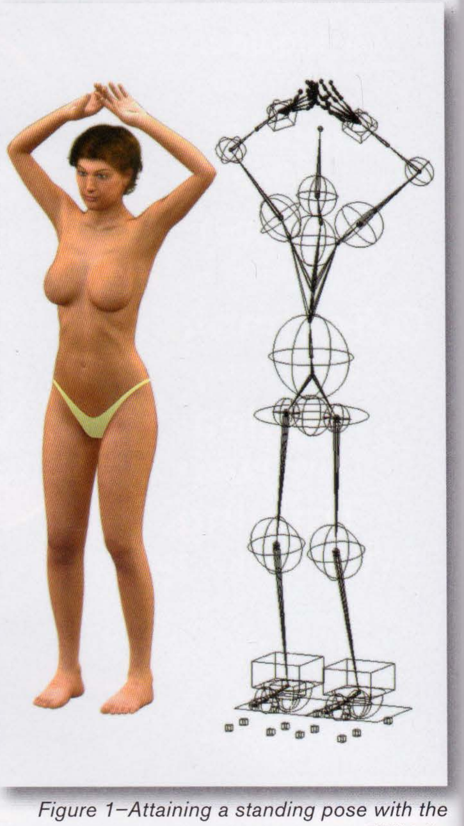


Figure 1–Attaining a standing pose with the FBIK rig.

STEP 5

Select the HipsEff effector and move it down so that the skeleton assumes a kneeling pose like the one in **Figure 2**. Since the LeftFootEff and RightFootEff effectors are pinned, the feet act like an anchor for the body. Rotate the feet back after selecting the LeftFootEff and RightFootEff effectors. Select the RightFootEffPivot and LefttFootEffPivot and rotate them. Select the LeftFootIndex1Eff and RightFootIndex1Eff effectors and move them back to position the toes.

STEP 6

Pin the HipsEff effector by selecting it, right-mouse-click, and pick Pin Both from the marking menu. This will pin both the translation and rotation of the hips effector. Notice that you also have the choice of pinning either translate or rotate individually. This is useful when you need to have some flexibility,

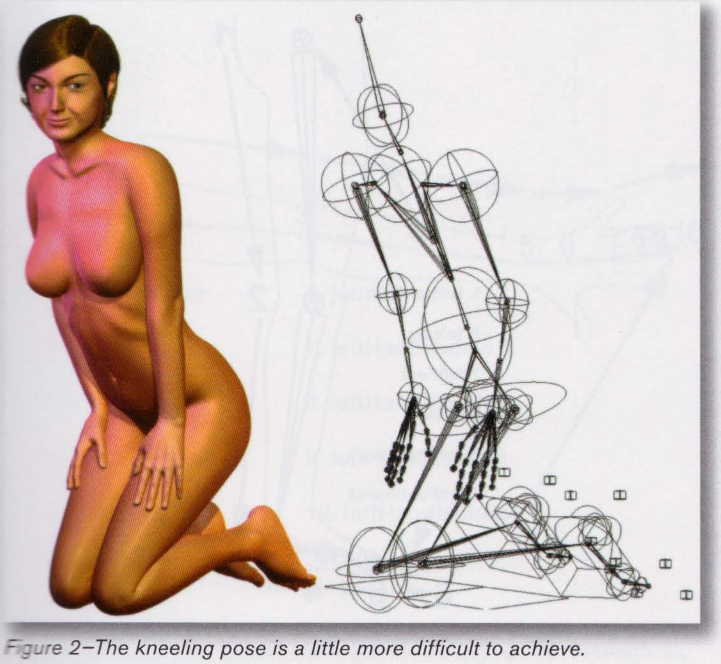


Figure 2–The kneeling pose is a little more difficult to achieve.

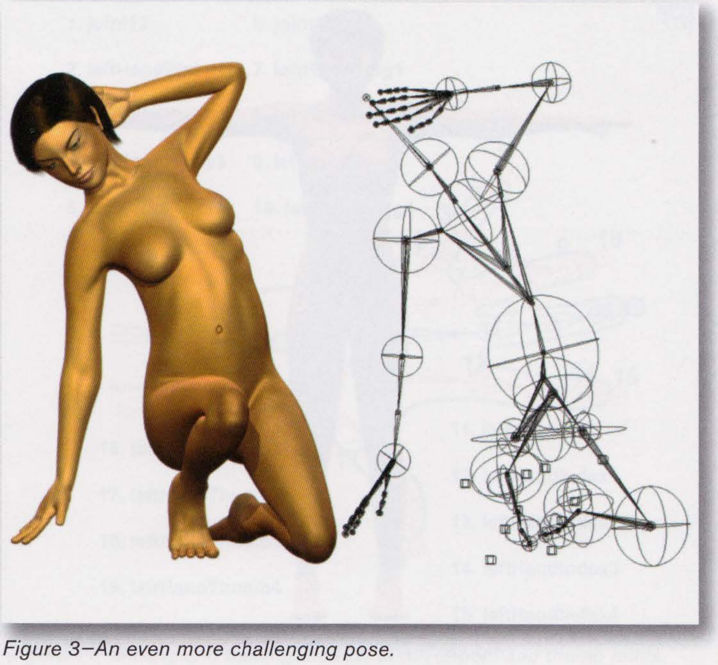


Figure 3–An even more challenging pose.

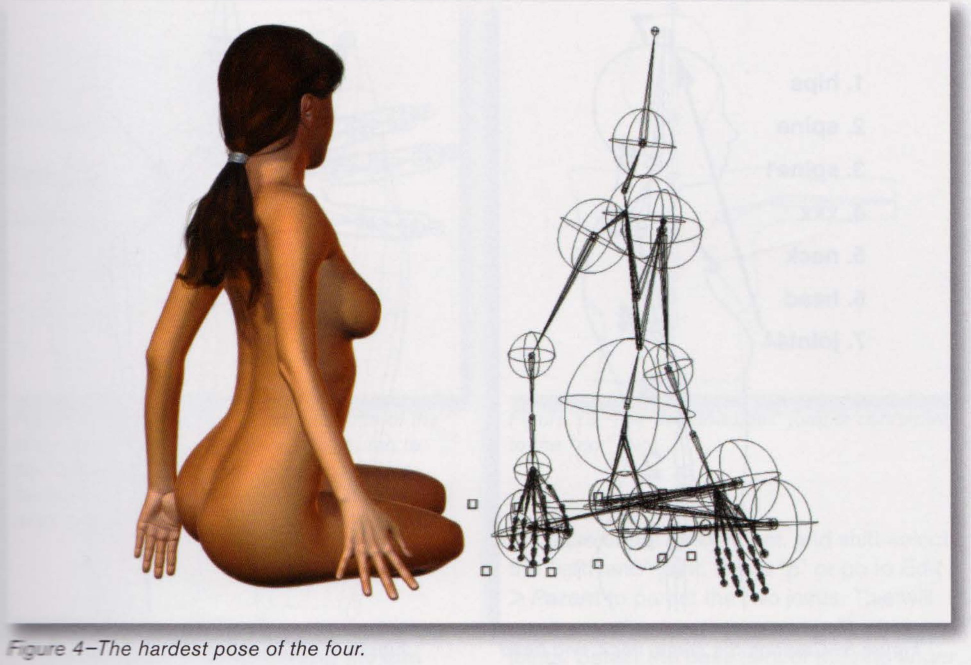


Figure 4–The hardest pose of the four.

such as allowing the effector to rotate when moving other effectors. Move the LefttHandEff and RightHandEff effectors to place the hands on the legs. Rotate and move other effectors until you achieve a pose similar to the one in **Figure 2**.

STEP 7

If you set keyframes for the second pose, then go to them on the timeline and right-click on each. Select ***Delete FBIK Keys > Delete All FBIK Keys***. Select a joint or an effector and go to ***Animation > Skeleton > Full Body IK > Go to Stance Pose***. Notice the option ***Animation > Skeleton > Full Body IK > Body Part Autoload***. Always keep this on because it will automatically load the character set that you are working with when you pick an effector. You no longer have to worry about choosing a character set first before setting keyframes.

STEP 8

Look at **Figure 3** and try to achieve this slightly more difficult pose. In order to establish a ground plane, select the LeftFootEff and RightFootEff effectors and right-mouse-click, then pick Unpin from the marking menu. While these effectors are still selected, go to ***Animation > Skeleton > Full Body IK > Add Floor Contact Plane***. Notice how the feet refuse to go below the Floor Contact Plane.

STEP 9

Unpin the HipsEff effector. Move the RightHandEff effector to the ground like the one in **Figure 3**. Notice that the fingers will not go below the Floor Contact Plane and even bend backwards. Continue moving and rotating effectors to achieve the third pose. You can pin the translation and/or rotation of specific effectors once you have them in the right position.

STEP 10

By now you should have had enough practice with the FBIK rig to attain the most difficult pose seen in **Figure 4**. Use the options that were discussed in the previous steps to reach this challenging posture.

PART 2: SETTING UP YOUR OWN FBIK RIG

Now it's time to create your own skeleton that fits your biped character. The first steps will have you use the joint tool and then label them correctly in accordance to FBIK specifications. The way you pose your character, the number of joints that you create, and the names that you assign to them are important if you don't want to get error messages.

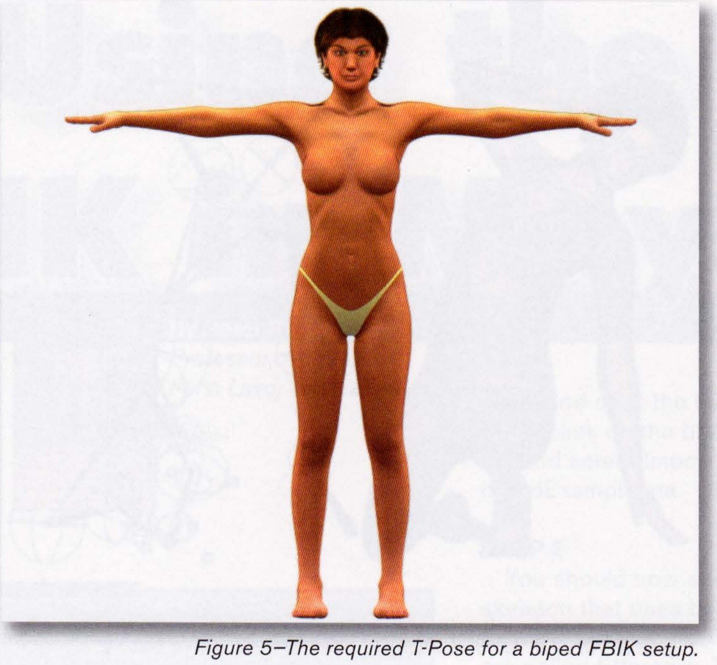


Figure 5–The required T-Pose for a biped FBIK setup.

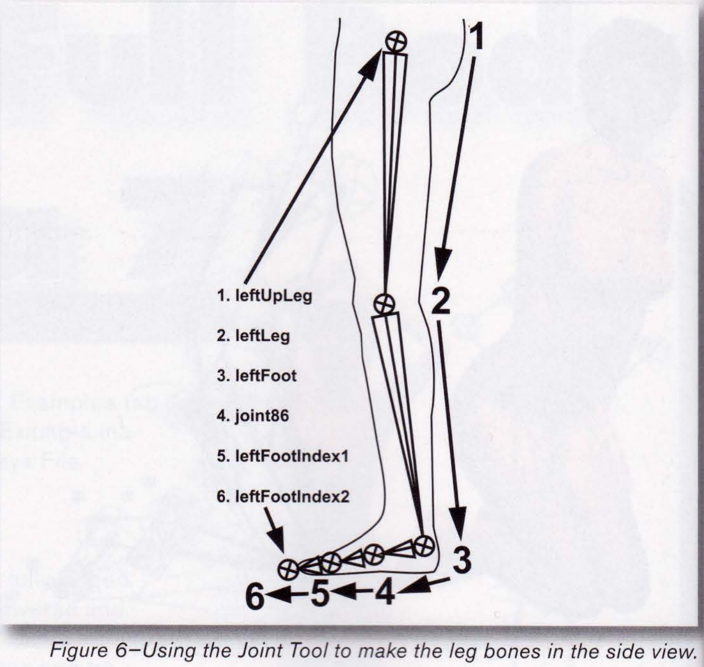


Figure 6–Using the Joint Tool to make the leg bones in the side view.

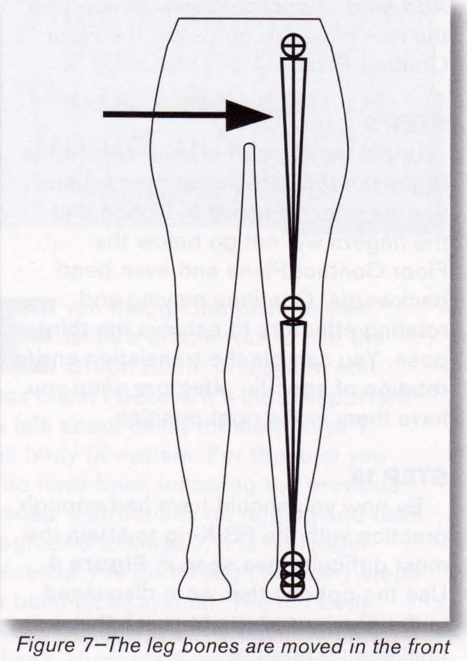


Figure 7–The leg bones are moved in the front view to the middle of the left leg.

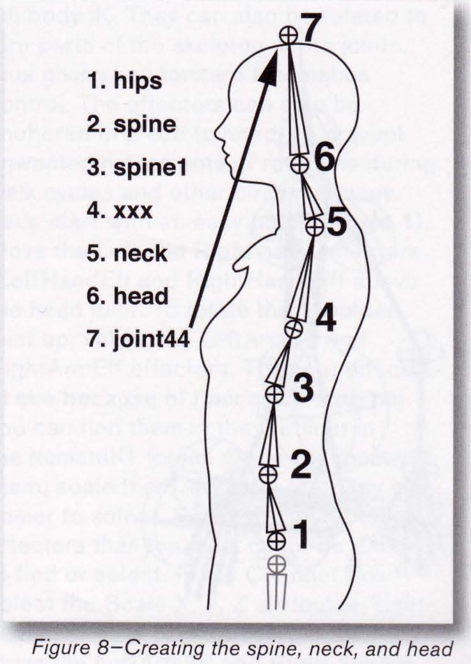


Figure 8–Creating the spine, neck, and head joints in the side view.

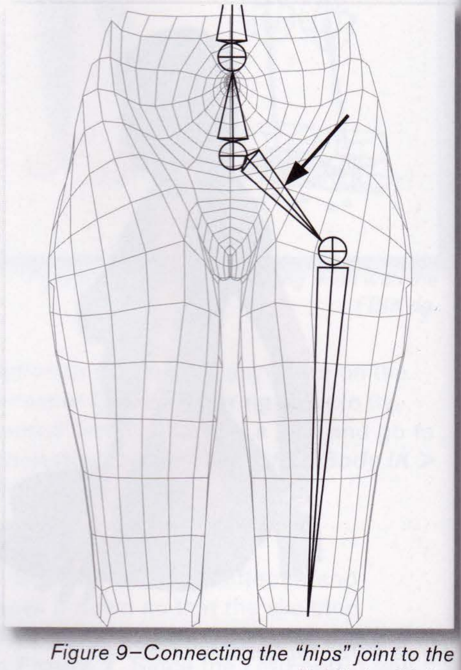


Figure 9–Connecting the "hips" joint to the "leftUpLeg" joint.

STEP 1

Position your character in the traditional T-Pose, like the one in **Figure 5**. The arms are straight out, palms of the hands should face down, and the legs are fairly straight up and down. Even though Maya's documentation says to do this, I had a student who bent the arms at the shoulder and armpits and had the legs bent at the knees and it still worked. She got an error message but it still created the FBIK rig.

STEP 2

Select the Joint Tool by going to ***Skeleton > Joint Tool***. Refer to **Figure 6** when you create and name each joint. In your side view, click with the tool at the hip area to make the "leftUpLeg" joint (1). If the first joint is too large or small, go to the Display menu and select Joint Size. Click at the knee to make the "leftLeg" joint (2). Click at the ankle for the "leftFoot" joint (3), the middle of the foot for the "joint86" (4), the "leftFootIndex1" (5), and finally in front of the toe to make the "leftFootIndex2" (6). Press Enter and in the Channel Box or the Outliner name each joint exactly like the ones in **Figure 6**.

STEP 3

Go to your front view, click on the top joint, and move the leg skeleton into the center of your character's left leg (**figure 7**).

STEP 4

In the side view, starting slightly above the hip joint, use the Joint Tool to create seven spine, neck, and head joints (**figure 8**). Rename the joints with the exact same names as seen in **Figure 8**.

STEP 5

Connect the "leftUpLeg" joint to the "hips" joint by selecting it first and then shift-selecting the "hips" joint. Press p or go to ***Edit > Parent*** to parent the leg to the hip. This will create a connector between the two pointing toward the "leftUpLeg" joint (**figure 9**).

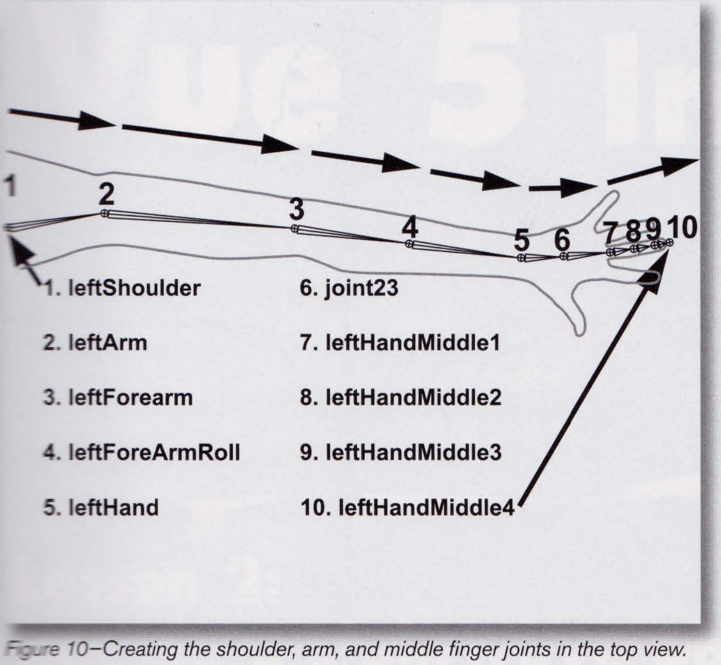

1
2
3
4
5
6
7
8
9
10
1. leftShoulder
2. leftArm
3. leftForearm
4. leftForeArmRoll
5. leftHand
6. joint23
7. leftHandMiddle1
8. leftHandMiddle2
9. leftHandMiddle3
10. leftHandMiddle4

Figure 10–Creating the shoulder, arm, and middle finger joints in the top view.

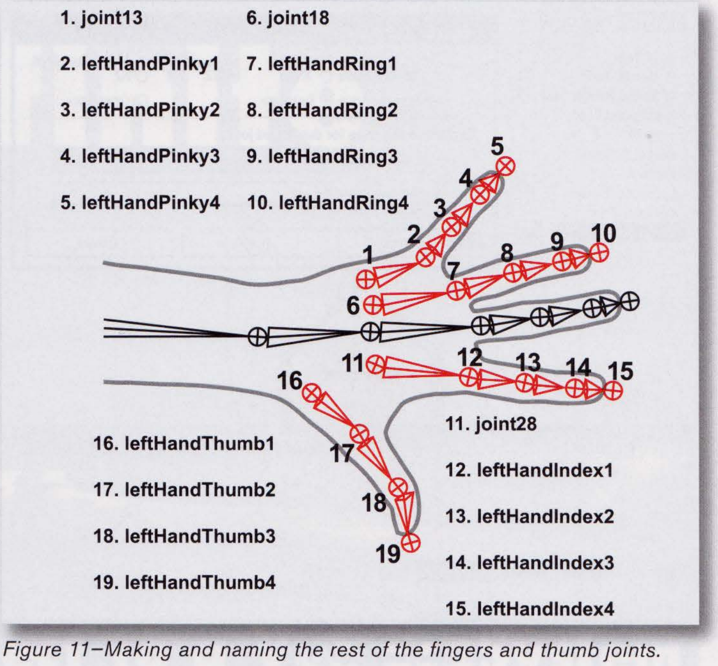

1. joint13
2. leftHandPinky1
3. leftHandPinky2
4. leftHandPinky3
5. leftHandPinky4
6. joint18
7. leftHandRing1
8. leftHandRing2
9. leftHandRing3
10. leftHandRing4
11. joint28
12. leftHandIndex1
13. leftHandIndex2
14. leftHandIndex3
15. leftHandIndex4
16. leftHandThumb1
17. leftHandThumb2
18. leftHandThumb3
19. leftHandThumb4

Figure 11–Making and naming the rest of the fingers and thumb joints.

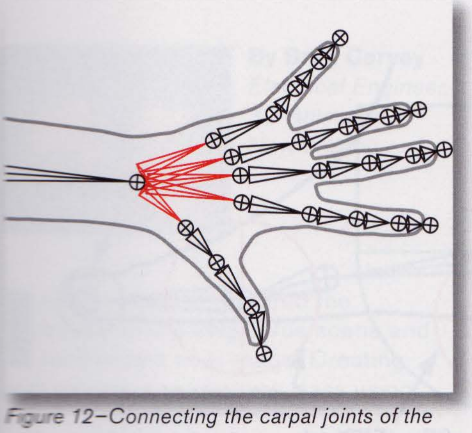


Figure 12–Connecting the carpal joints of the pinky, ring finger, index finger, and thumb to the "leftHand" joint by parenting them. Note the new connecting bones that extend from the wrist joint toward all the fingers and the thumb.

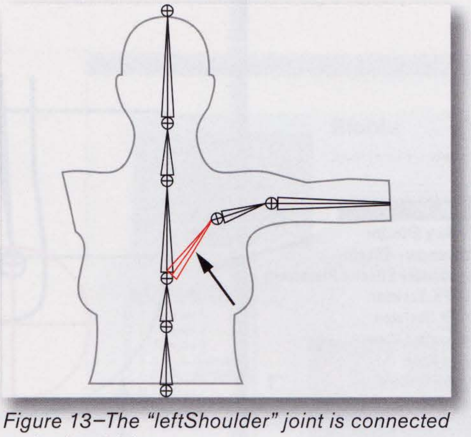


Figure 13–The "leftShoulder" joint is connected to the "xxx" joint.

STEP 6

The next step involves making the arm and hand joints. Switch to the top view and select the Joint Tool. Starting at the left shoulder, create a series of joints similar to the ones shown in **Figure 10**. The last joint is at the end of the middle finger. Name the joints exactly as shown in **Figure 10**.

STEP 7

Make the pinky, ring finger, index finger, and thumb joints. Create these groups separately. In other words, when you finish one finger, click or select the Joint tool again to start a new set of joints for the next finger. Refer to **Figure 11** for the exact names of each joint.

STEP 8

The pinky, ring finger, index finger, and thumb will now be connected to the "leftHand" joint. Select "joint13," which is the base of the pinky finger, and shift-select the "leftHand" joint. Press "p" or go to ***Edit > Parent*** to parent the two joints. This will create another bone connecting the two joints. Select the base joint of the ring finger ("joint18") and shift-select the "leftHand" joint. Press "p" to parent these two. Parent the index finger and thumb to the "leftHand" joint. Your hand bones should now look somewhat like the ones in **Figure 12**, with the connectors from the "leftHand" joint displayed in red.

STEP 9

Use your front and/or side views to move the entire arm and hand skeleton up so that it is inside the arm of your model. Adjust the hand and finger joints by moving and rotating them until each is inside your character's hand.

STEP 10

Select the "leftShoulder" joint. Shift-select the joint named "xxx." Press p to parent so that you now have a connector similar to the red one shown in **Figure 13**.

STEP 11

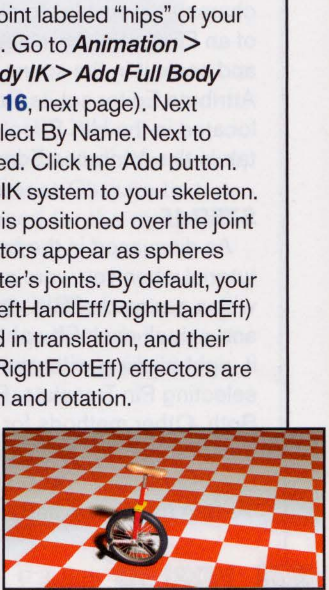
Click on the "leftUpLeg" joint and go to ***Skeleton > Mirror Joint > Options***. In the Mirror Joint Options box, for Mirror Across, select the YZ axes. Next to "Search For:" type "left." For "Replace With:" type "right." Click the Mirror button. You should now have a skeleton for the right leg that is connected to the "hips" joint (**figure 14**, next page).

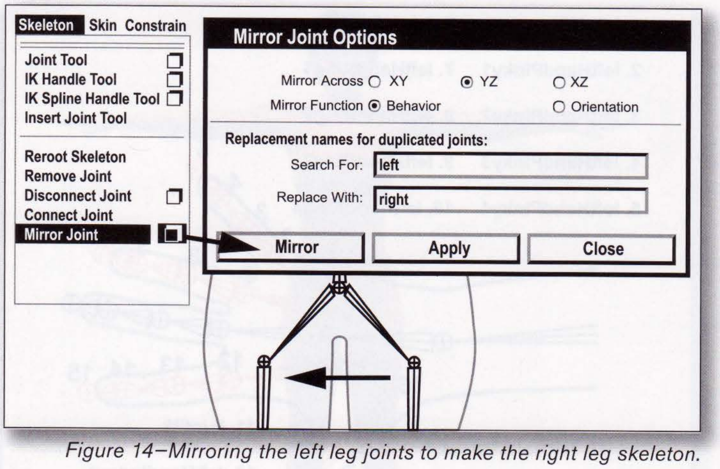
STEP 12

Select the "leftShoulder" joint and go to ***Skeleton > Mirror Joint > Options***. In the Mirror Joint Options box, for Mirror Across, select the YZ axes. Next to "Search For:" type left. For "Replace With:" type right. Click the Mirror button. The arm should mirror across the X-axis and automatically connect to the "xxx" joint (**figure 15**, next page).

STEP 13

Select the Root joint labeled "hips" of your completed skeleton. Go to ***Animation > Skeleton > Full Body IK > Add Full Body IK Options*** (**figure 16**, next page). Next to Identify Joints, select By Name. Next to Posture, select Biped. Click the Add button. This will add the FBIK system to your skeleton. Each FBIK effector is positioned over the joint it drives. FBIK effectors appear as spheres around your character's joints. By default, your character's wrist (LeftHandEff/RightHandEff) effectors are pinned in translation, and their ankle (LeftFootEff/RightFootEff) effectors are pinned in translation and rotation.




Skeleton
Skin
Constrain
Joint Tool ☐
IK Handle Tool ☐
IK Spline Handle Tool ☐
Insert Joint Tool
Reroot Skeleton
Remove Joint
Disconnect Joint ☐
Connect Joint
Mirror Joint ☐
Mirror Joint Options
Mirror
Apply
Close

Mirror Across ○ XY ● YZ ○ XZ

Mirror Function ● Behavior ○ Orientation

Replacement names for duplicated joints:

Search For: left

Replace With: right

Figure 14–Mirroring the left leg joints to make the right leg skeleton.

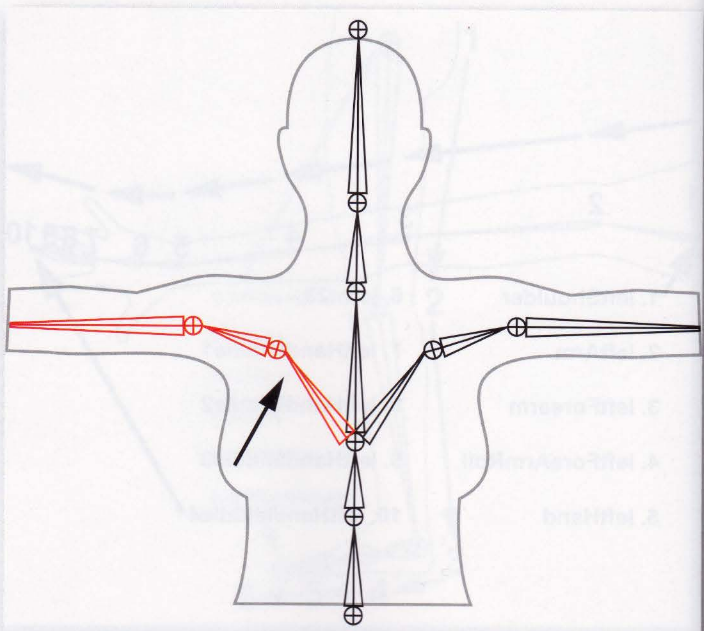


Figure 15–The mirrored right arm joints.

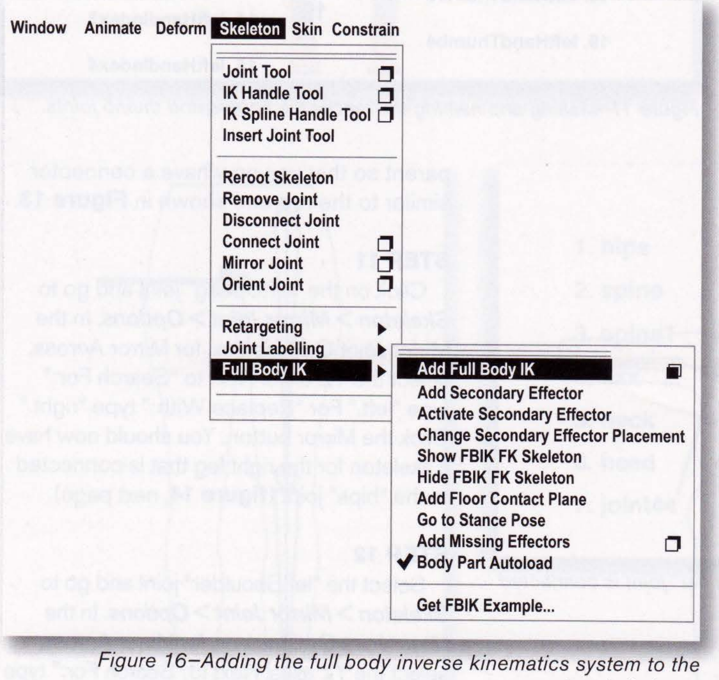

Window
Animate
Deform
Skeleton
Skin
Constrain
Joint Tool ☐
IK Handle Tool ☐
IK Spline Handle Tool ☐
Insert Joint Tool
Reroot Skeleton
Remove Joint
Disconnect Joint
Connect Joint ☐
Mirror Joint ☐
Orient Joint ☐
Retargeting
Joint Labelling
Full Body IK
Add Full Body IK ☐
Add Secondary Effector
Activate Secondary Effector
Change Secondary Effector Placement
Show FBIK FK Skeleton
Hide FBIK FK Skeleton
Add Floor Contact Plane
Go to Stance Pose
Add Missing Effectors ☐
✓ Body Part Autoload
Get FBIK Example...

Figure 16–Adding the full body inverse kinematics system to the completed skeleton.

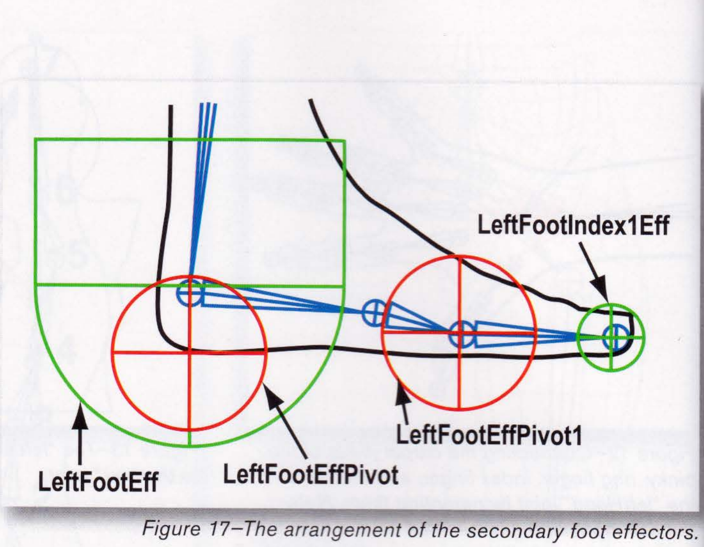

LeftFootIndex1Eff
LeftFootEffPivot1
LeftFootEff
LeftFootEffPivot

Figure 17–The arrangement of the secondary foot effectors.

STEP 14

Scale each Eff sphere so that you can easily see and select them when your character is visible. To change the size of an FBIK effector, select the effector and scale it in the scene view, or in the Attribute Editor set its Radius attribute located in the HIK Effector's JointNameEff tab in the Attributes Editor.

STEP 15

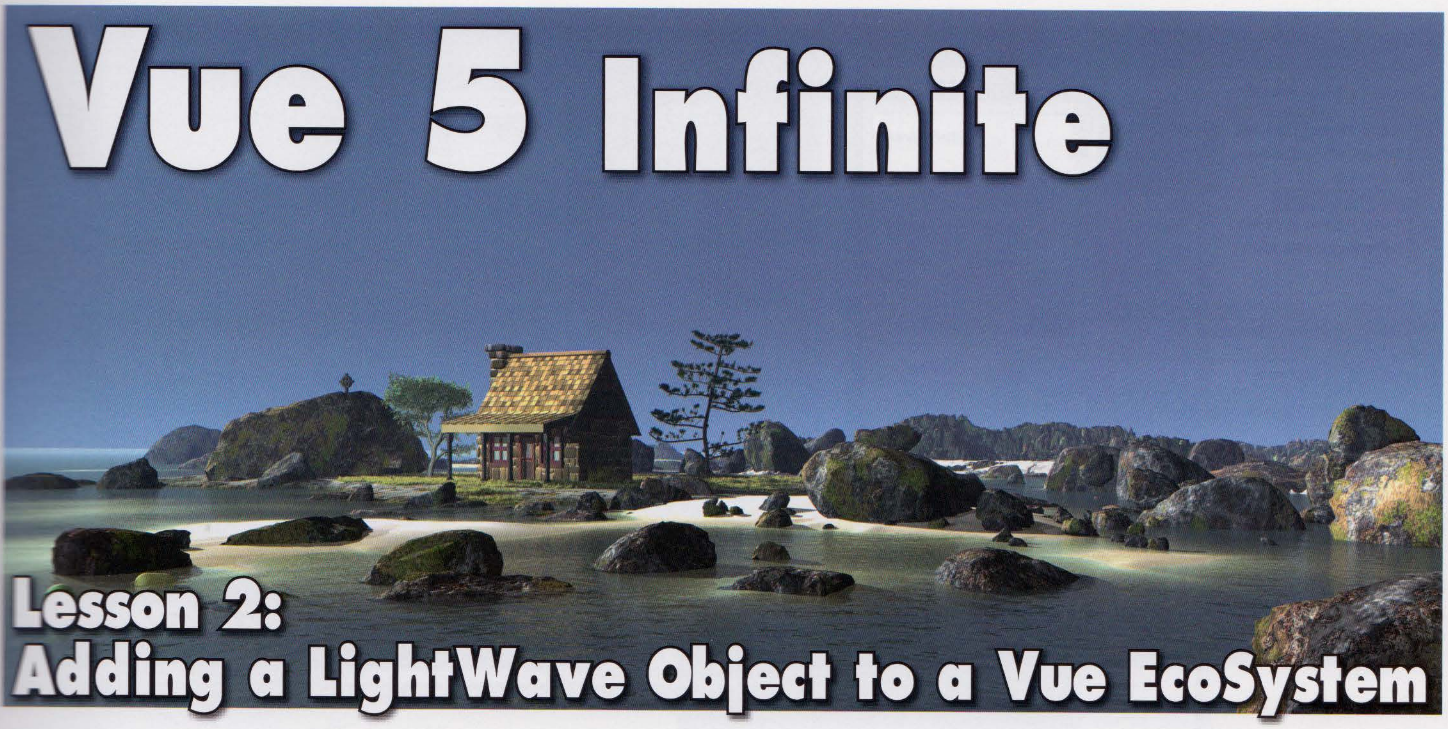
As discussed in the beginning of this tutorial when you were experimenting with a pre-made FBIK rig, you can lock and unlock each Eff sphere by selecting it, right-clicking with your mouse, and selecting Pin Translate, Pin Rotate, or Pin Both. Other methods for pinning are from the Channel Box and by using the Alt+w (pin translation), Alt +e (pin rotation), or Alt +s (pin translation and rotation) hotkeys.

STEP 16

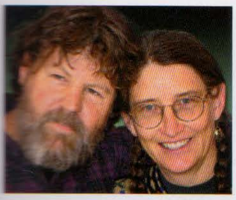
Some of you may want to set up secondary effectors for actions such as a walk in which the foot rolls. The following steps show how to create secondary effectors for the heel and ball of the foot. Select the LeftFootEff FBIK effector. Go to ***Animation > Skeleton > Full Body IK > Add Secondary Effector***. Move it to the heel of your person's left foot. Scale it up if you need to see it more clearly. While the LeftFootEffPivot is still selected, go to ***Animation > Skeleton > Full Body IK > Activate Secondary Effector***. Now when you rotate the LeftFootEff effector, it will revolve around the secondary effector named "LeftFootEffPivot." Create another secondary effector for the ball of the foot. Select the LeftFootEff and go to ***Animation > Skeleton > Full Body IK > Add Secondary Effector***. Move the newly created LeftFootEffPivot1 to the ball of the left foot. Activate the LeftFootEffPivot1 by going to ***Animation > Skeleton > Full Body IK > Activate Secondary Effector***. **Figure 17** shows the newly created foot roll effectors and their placement.

***Peter Ratner** is a professor of 3D Computer Animation at James Madison University. He is the founder and head of the first Computer Animation program in Virginia. His paintings, animations, and computer graphics have been displayed in numerous national and international juried exhibitions. He is the author of* 3-D Human Modeling and Animation*, 1st and 2nd Editions (John Wiley and Sons), and* Mastering 3D Animation*, 1st and 2nd Editions (Allworth Press). He lives in Penn Laird, Virginia.*

Vue 5 Infinite



Lesson 2:
Adding a LightWave Object to a Vue EcoSystem



By Brad Carvey
Electrical Engineer,
3D Animator
New Mexico

In Lesson 1, we explored the basics of creating a Vue scene and rendering a cool image. Creating cool images was relatively easy using Vue EcoSystems.

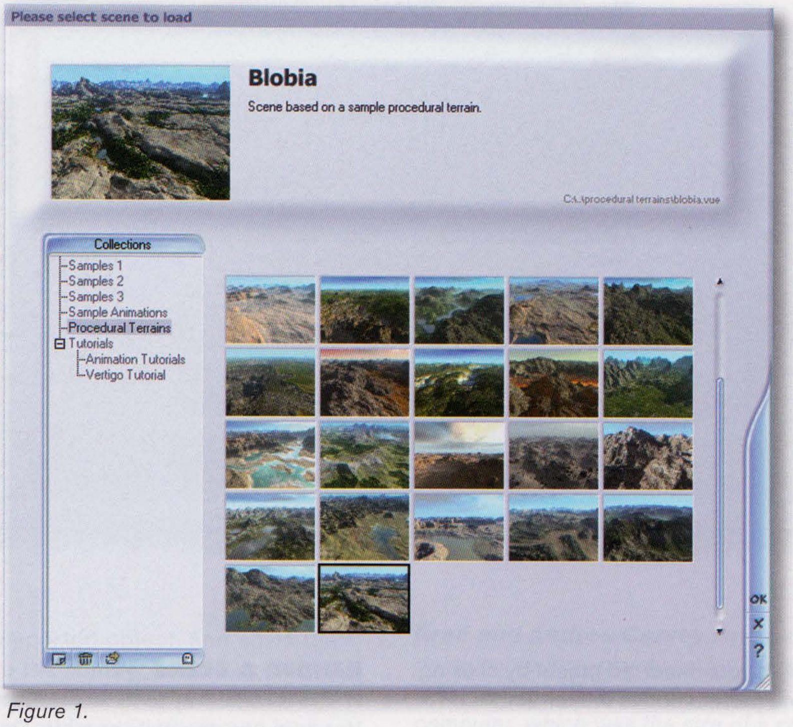
Now that we can create cool images, we need to learn how to add a LightWave object to the Vue scene. Let's get started by loading one of the sample scenes that comes on the Vue Extras CD. You will need to copy the scene folder from the CD to the Vue scene folder to access the scenes that I chose for this article.

Remember that I am learning this program at the same time as you are. As a LightWave artist I am learning and experimenting with Vue, but I am not an expert. My lack of expertise is useful because I am figuring out how a LightWave user would find this useful.

LOADING A VUE SCENE

You can load any scene that you like. I chose a procedural terrain called Blobia. The interesting thing about procedural terrains is that you can zoom in as close as you want and they will still look good. Try moving the camera around the terrain and rendering scenes with the camera very close to the terrain.

1. Run Vue and click on Open (Ctrl + O).



Please select scene to load

Blobia

Scene based on a sample procedural terrain.

C:\..\procedural terrains\blobia.vue

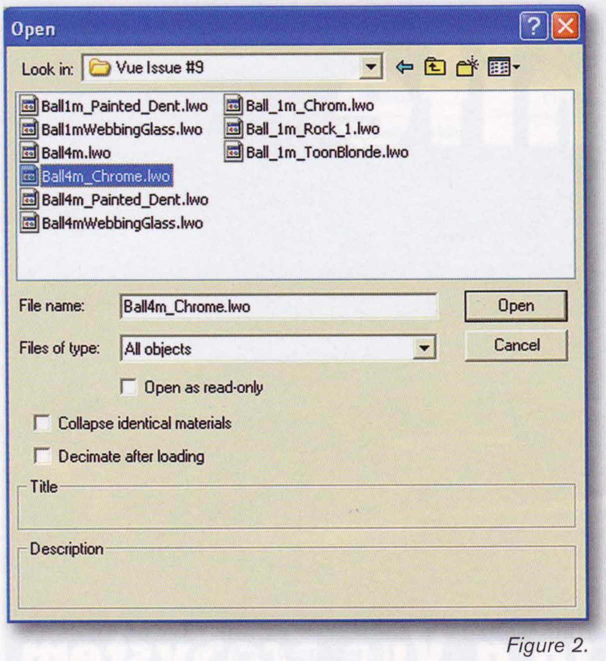
Figure 1.

2. Select Procedural Terrains, scroll down, and click on image for Blobia. (**figure 1**).

ADDING A LIGHTWAVE OBJECT

Vue does not work on real world units. In other words, the size of things in Vue appears to be random. I imported a 1-meter ball and a 4-meter ball into the Blobia scene. Both objects appeared to be hundreds of meters high, and both were the same size. Only basic LightWave Object textures imported correctly. For example, I could not get bump maps to import and render in Vue, but LightWave's preset Chrome texture worked and was used for the rendered images in this article.

1. Click on Import Object and select any LightWave object. I created a 4-meter ball that I imported (**figure 2**, next page).



Open

Look in: Vue Issue #9

Ball1m_Painted_Dent.lwo
Ball1mWebbingGlass.lwo
Ball4m.lwo
Ball4m_Chrome.lwo
Ball4m_Painted_Dent.lwo
Ball4mWebbingGlass.lwo
Ball_1m_Chrom.lwo
Ball_1m_Rock_1.lwo
Ball_1m_ToonBlonde.lwo

File name: Ball4m_Chrome.lwo

Files of type: All objects

☐ Open as read-only

☐ Collapse identical materials

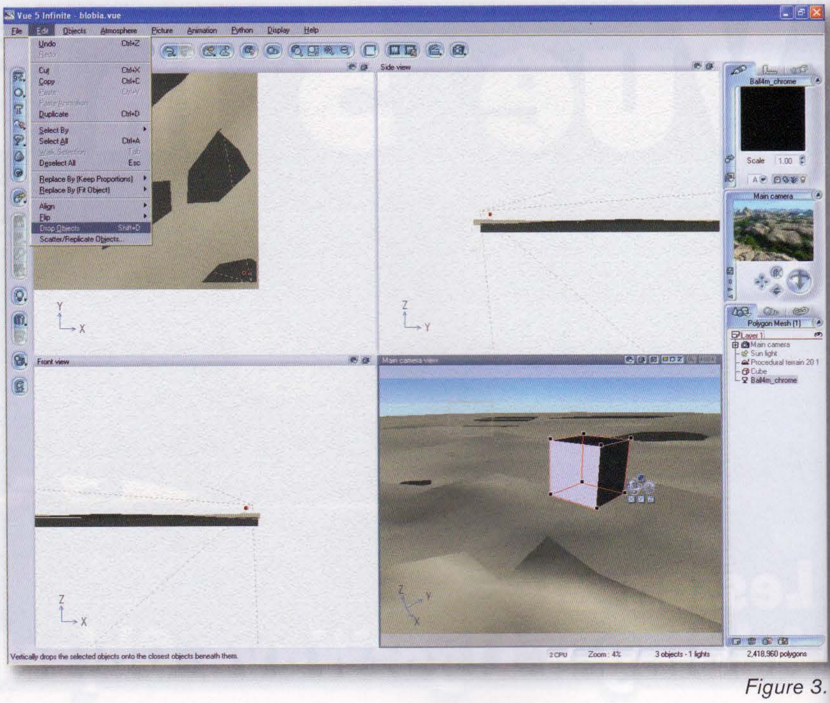
☐ Decimate after loading

Title

Description

Open
Cancel

Figure 2.



Vue 5 Infinite - blobia.vue
File
Edit
Objects
Atmosphere
Picture
Animation
Python
Display
Help
Undo
Redo
Cut
Copy
Paste
Duplicate
Select By
Select All
Deselect All
Replace By (Keep Proportions)
Replace By (Fit Object)
Align
Flip
Drop Objects
Scatter/Replicate Objects...
Side view
Front view
Main camera view
Ball4m_chrome
Main camera
Polygon Mesh (1)
Layer 1
Main camera
Sun light
Procedural terrain 20 1
Cube
Ball4m_chrome
Vertically drops the selected objects onto the closest objects beneath them.
2 CPU
Zoom : 4%
3 objects - 1 lights
2,418,960 polygons

Figure 3.

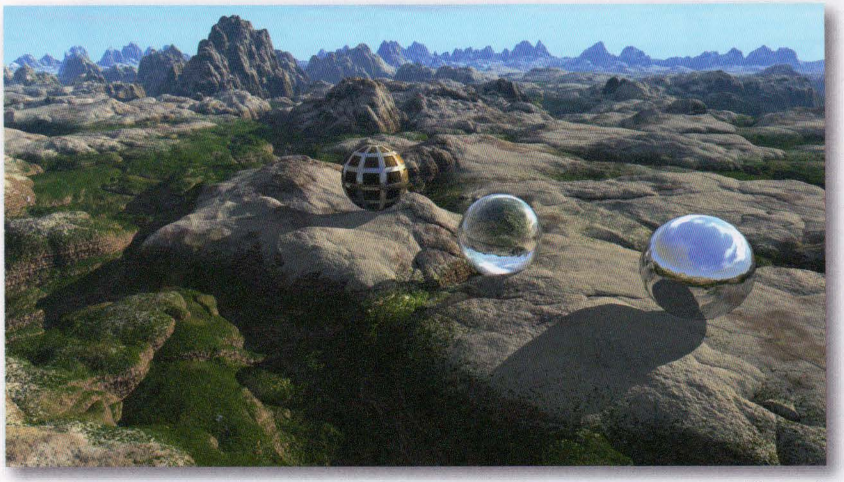
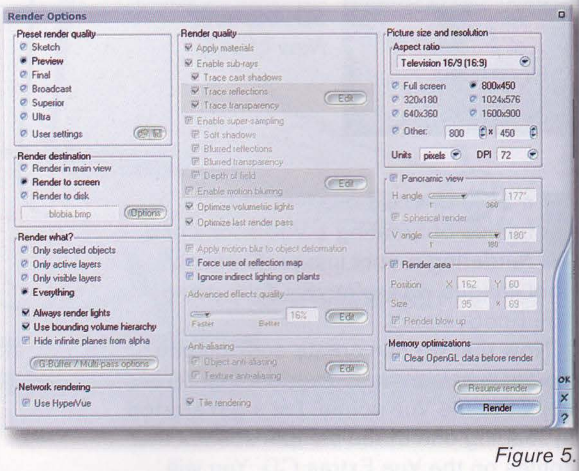


Figure 4.



Render Options

Preset render quality
○ Sketch
● Preview
○ Final
○ Broadcast
○ Superior
○ Ultra
○ User settings

Render destination
○ Render in main view
● Render to screen
○ Render to disk
blobia.bmp

Render what?
○ Only selected objects
○ Only active layers
○ Only visible layers
● Everything
☑ Always render lights
☑ Use bounding volume hierarchy
☐ Hide infinite planes from alpha

Network rendering
☐ Use HyperVue

Render quality
☑ Apply materials
☑ Enable sub-rays
☑ Trace cast shadows
☑ Trace reflections
☑ Trace transparency
☐ Enable super-sampling
☐ Soft shadows
☐ Blurred reflections
☐ Blurred transparency
☐ Depth of field
☐ Enable motion blurring
☑ Optimize volumetric lights
☑ Optimize last render pass
☐ Apply motion blur to object deformation
☐ Force use of reflection map
☐ Ignore indirect lighting on plants

Advanced effects quality
Faster Better 16%

Anti-aliasing
☐ Object anti-aliasing
☐ Texture anti-aliasing

☑ Tile rendering

Picture size and resolution
Aspect ratio
Television 16/9 (16:9)
○ Full screen
● 800x450
○ 320x180
○ 1024x576
○ 640x360
○ 1600x900
○ Other: 800 × 450
Units pixels DPI 72

☐ Panoramic view
H angle 1 360 177°
☐ Spherical render
V angle 1 180 180°

☐ Render area
Position X 162 Y 60
Size 95 × 69
☐ Render blow up

Memory optimizations
☐ Clear OpenGL data before render

Options
G-Buffer / Multi-pass options
Edit
Edit
Edit
Edit
Resume render
Render
OK
X
?

Figure 5.

2. Select your imported object by clicking on it and move it above the terrain.

3. Click on Drop Object (Ctrl + D) to drop the object onto the terrain (**figure 3**).

You can scale the object before dropping it, and you can place the object on the terrain, but the easiest thing to do, at first, is to just drop it. I imported and dropped a few balls on the Blobia terrain. The glass and chrome balls were imported with LightWave textures that were selected from metal and glass presets. I assigned a Vue Material to the third ball and rendered the scene (**figure 4**).

RENDER A SCENE

The easiest way to render a scene in Vue is to use one of its presets. Depending on how much time you have, any preset up to "Ultra" can be used. It is also possible to modify the presets and create a custom rendering profile. I decided to render to the screen and use a 16:9 aspect ratio.

1. Select *Picture > Render Options* (**figure 5**).

2. Select a Preset render quality; I chose Ultra.

3. Select *Render to screen.*

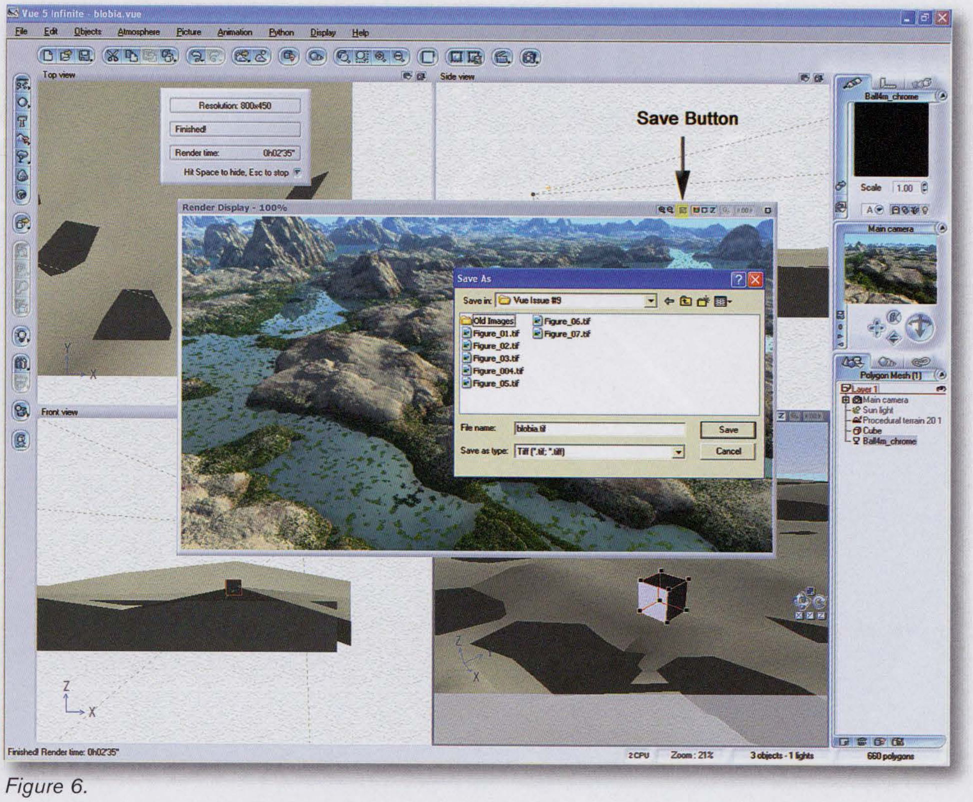
4. Select *Television 16/9* from the Aspect Ratio drop-down menu.

5. Select the resolution that you would like to render.

6. Click on Render.

7. After the Render is finished and if you like the results, click on the save image icon at the right top of the image and save the image (**figure 6**).

I like rendering to an image, so I can see the image and decide to abort if I don't like the image and save the image if I do like it. After successfully rendering a Vue Scene with a LightWave object, you should experiment with loading more


Vue 5 Infinite - blobia.vue
File
Edit
Objects
Atmosphere
Picture
Animation
Python
Display
Help
Top view
Side view
Front view
Save Button
Render Display - 100%
Save As
Old Images
Figure_01.tif
Figure_02.tif
Figure_03.tif
Figure_004.tif
Figure_05.tif
Figure_06.tif
Figure_07.tif
Save
Cancel
Ball4m_chrome
Main camera
Polygon Mesh (1)
Layer 1
Main camera
Sun light
Procedural terrain 20 1
Cube
Ball4m_chrome
Z
X

Resolution: 800x450

Finished!

Render time: 0h02'35"

Hit Space to hide, Esc to stop

Scale 1.00

Save in: Vue Issue #9

File name: blobia.tif

Save as type: Tiff (*.tif; *.tiff)

Finished! Render time: 0h02'35" 2 CPU Zoom : 21% 3 objects - 1 lights 660 polygons

Figure 6.

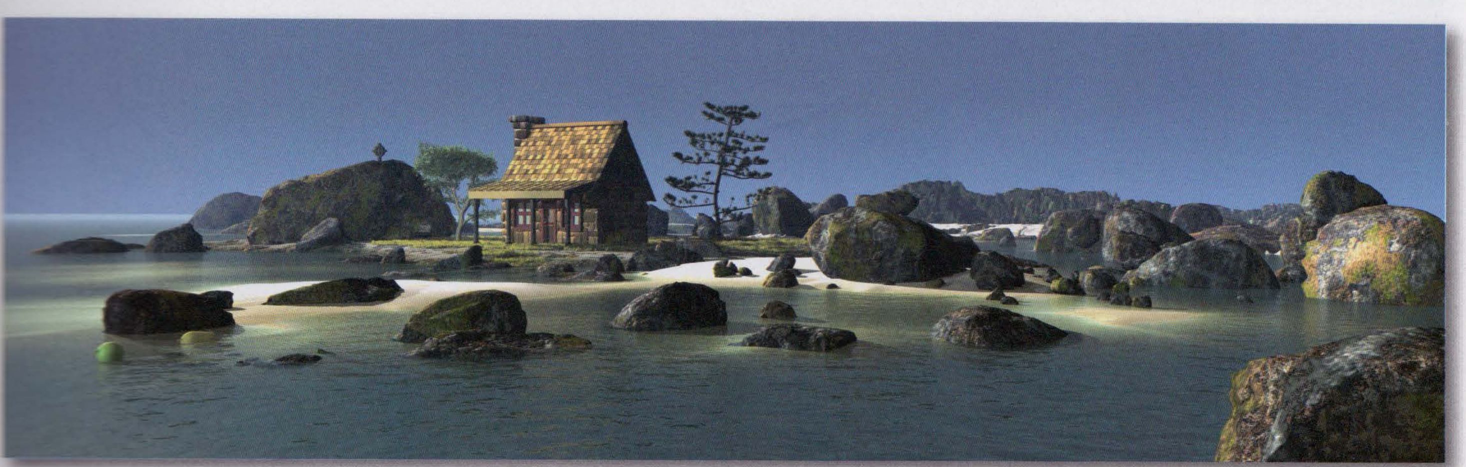


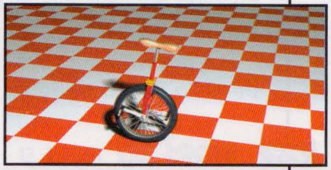
Figure 7.

complex objects. I loaded a desert island scene and added a small house object that comes with LightWave (**figure 7**). The object had lots of complex textures, but only the basic colors were rendered in Vue.

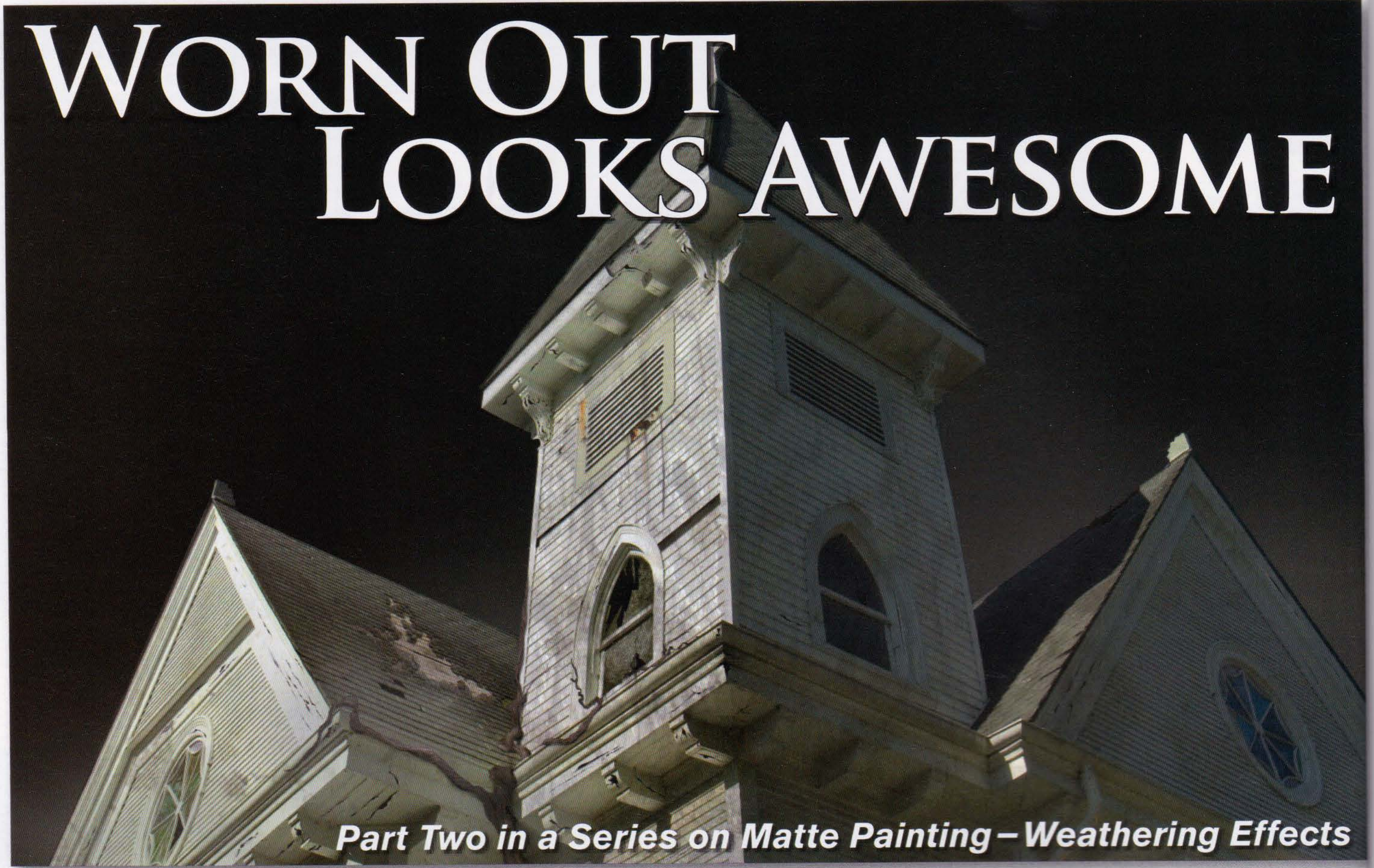
After playing around with importing LightWave objects, there are a few things that appear to be problems. First, you will need to scale your objects after importing them. It may be tricky to get the relative sizes of objects to be correct. It might be a good idea to model a stick with alternating colors every meter. Import that into the scene and scale it until it looks like the meter long colors are the right size in your scene. Then import your other objects. Place the scaled measuring stick next to your imported object, and scale the object to the number of meters it should be. If you do this with every imported object, all of the objects should look like they are the correct size relative to themselves and to the terrain.

The second problem is texturing. After playing around with Vue, I discovered that you could select a texture on an imported LightWave object and edit the texture in Vue. The Vue material editor has lots of presets and is very powerful.

***Brad and Andrea Carvey** have been doing computer animations for a long time. In 1969, Brad used an analog computer, which was the size of a car, to produce his first computer animation. Andrea, an archeologist, prefers to do scientific animations. Her credits include programs like The Discovery Channel's* Understanding Cars. *Brad is an electrical engineer and an Emmy Award-winning member of the Video Toaster development team. He prefers to do feature film work. His credits include films like* Men in Black, Stuart Little, Black Hawk Down, Kate & Leopold, *and* Master of Disguise.



WORN OUT LOOKS AWESOME



Part Two in a Series on Matte Painting–Weathering Effects

By Briar Lee Mitchell
Animator, Illustrator, Teacher
Los Angeles, California

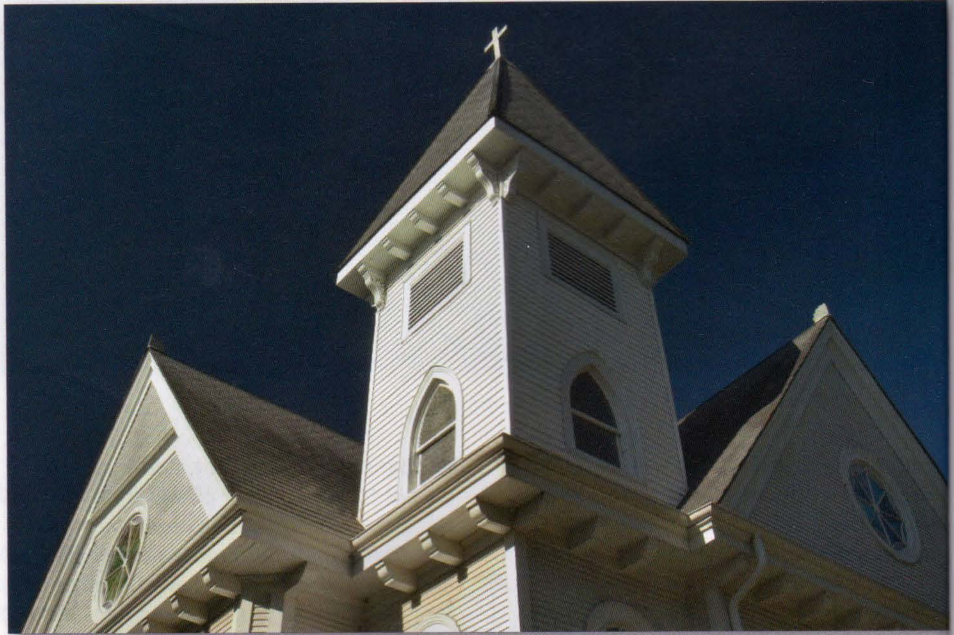


Figure 1–Trinity Evangelical Lutheran Church (c. 1911) in Algiers Point, New Orleans, Louisiana, May 1, 2005. Photographer–Alexey Sergeev.

I love to work on my home here in Southern California–spackling, painting, landscaping. I'm a fiend for all that stuff because I like my house to look nice, but nothing makes me happier in matte painting than to take a pristine building and destroy it.

I found this wonderful photo by Alexey Sergeev of a gorgeous old church in New Orleans that has been, as you can see, kept in superb condition (**figure 1**). Let's wear it out.

Before I begin to wear out the building, I separate the structure from the background so I can paint or place a new sky back there–or you can composite the painting with moving footage for your media projects. To complete the effect I am going for, I want to show off the church at night, so I painted a darker less colorful sky and added a tiny bit of fog and light to help silhouette the structure. Separating the building out also allows me to clip textures to it (**figure 2**).

Anyone who has lived in a wood building like this knows that it needs to be painted regularly to keep up its condition and appearance. This building is around 100 years old, so as I was imagining what that paint would look like peeling off the structure, I felt that it would have several layers that could crack and curl away from the wood like an orange peel, instead of thinner coats of paint that tend to peel away and flutter like paper. Anytime I take an existing image like this and alter it to fit "the story," in this case, taking a beautiful church and turning it into a worn, spooky structure, I do some research. What materials am I working with? How old is the building? How much damage and/or atmosphere do I need to inject into the piece? When paint peels or cracks away

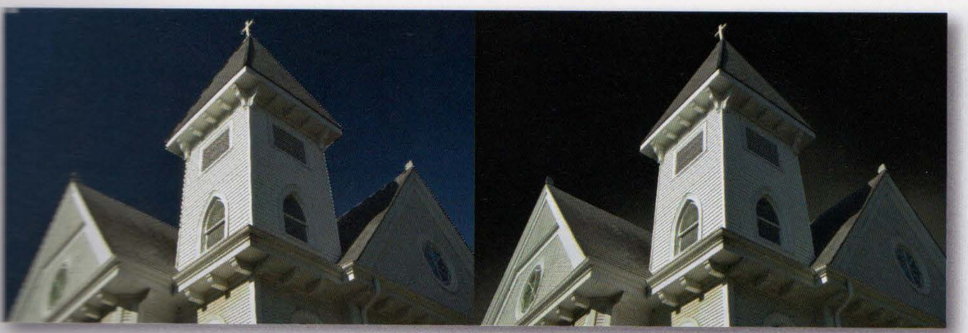


Figure 2–Separate the building onto its own layer so you can drop in skies behind it and clip textures cleanly to it.

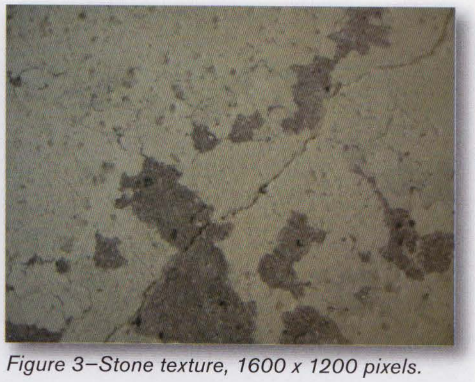
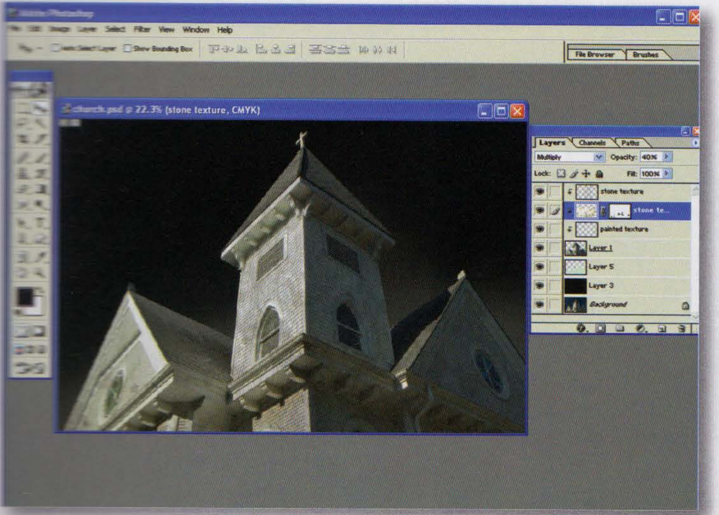


Figure 3–Stone texture, 1600 x 1200 pixels.



Multiply

Opacity: 40%

Fill: 100%

Figure 4–Stone texture has been clipped to the building, multiplied, and the layer reduced in opacity.

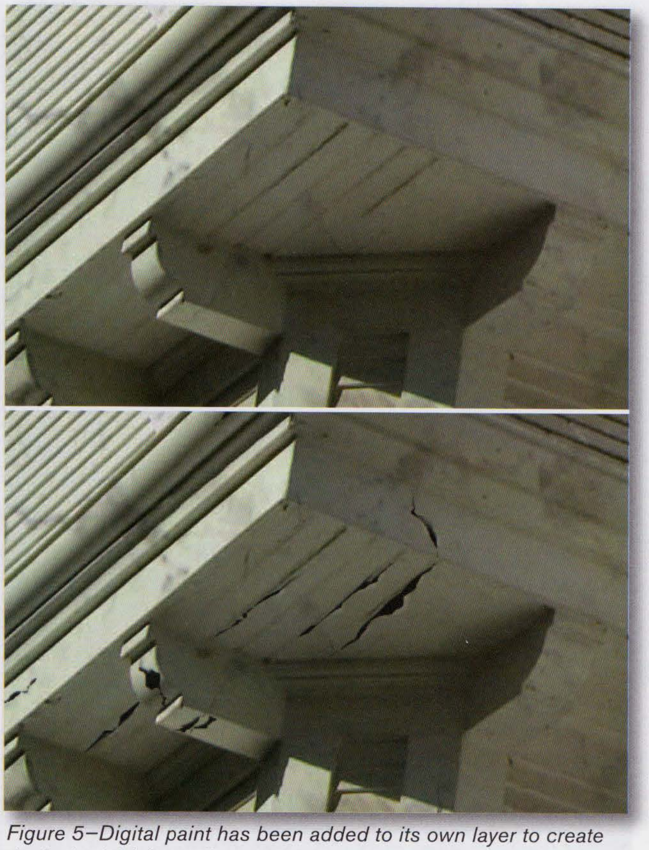


Figure 5–Digital paint has been added to its own layer to create the look of thick, peeling paint.

from the exposed wood it can splinter, turn a ghastly shade of gray, or grow mold, whereas the metal vents and pipes tend to rust and cause streaking and so on. Observe the world around you. All the answers you need are there.

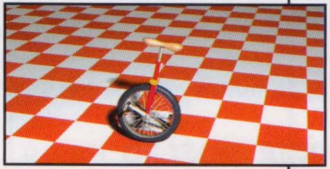
I try to avoid painting on existing images as much as possible to achieve the effects I am going for. The minute I start to paint on photos, I am wiping out great texture that is already there, and the piece can start to take on an artificial feeling. Wherever possible I find ways to manipulate the photo using levels, hue/saturation, or curves, or adding other elements. To create the aged appearance of the white paint, I found a great image of existing texture to work with, in this case, stone (**figure 3**). Stone has a wonderful organic feel to it. When you use existing textures like this, you want to avoid stretching or sizing the texture up, as it will degrade, so start with an image that is large enough to comfortably cover the areas you need.

Place the texture over the building and clip it into place, then multiply the layer. All of the white will drop away, leaving the cool looking splotches and cracks that quickly give the paint a worn feeling and give the appearance of mold, stains, and other damage. Adjust the opacity of the layer so that it blends with the existing image. I highly recommend you use a layer mask to then subtract the texture away from certain areas of the building, like the windows or sharp angles like under the eaves.

I paint streaks on the building that can appear on areas where water will naturally accumulate, pulling sediment and pigment that leeches out of wood or rust streaks from the metal. Paint can also yellow and get quite dingy looking with age so you may want to adjust the color of the stone texture a bit to add that effect. Cut apart and move the texture as much as you need to cover the building (**figure 4**).

Where I want to make the paint look like it is peeling, I digitally paint a dark black/green color for the shadow areas, then sample the original paint color of the building and lighten that up a bit for some highlights. Take some time to research other images here. Walk around town, look at photos in books or on the web of similar painted wood structures that have the same kind of damage you are going for. You will notice certain areas, like the exposed rounded ends of the roof supports, that peel broadly in a circular pattern, whereas flat areas that meet at 90-degree angles tend to peel away in sheets. I look for other areas on the building where there is a bit of peeling already there and enhance that with my paint techniques. The more I can work with what is already there, the more realistic my final effects will be (**figure 5**).

To wear the roof out, I used a photo of a damaged roof and harvested the areas of exposed plywood and damaged shingles to add to the roof of the church I am altering. Once I placed the image on the roof, I used the distort feature to ease it into the right



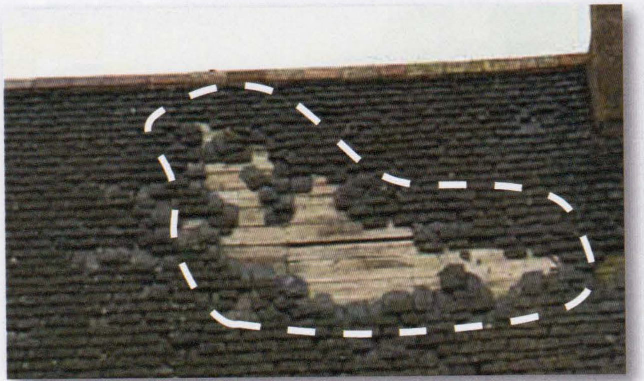


Figure 6a–The dashed lines indicate the area I am going to harvest and place on the roof of the church I am weathering.

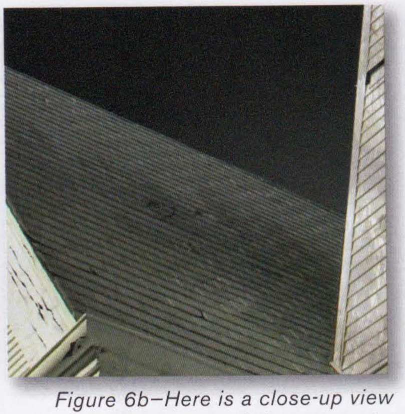


Figure 6b–Here is a close-up view where I want to add damage to the church's roof.

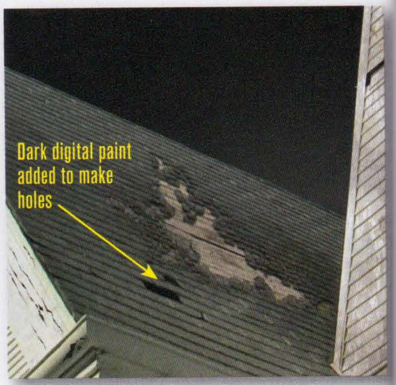
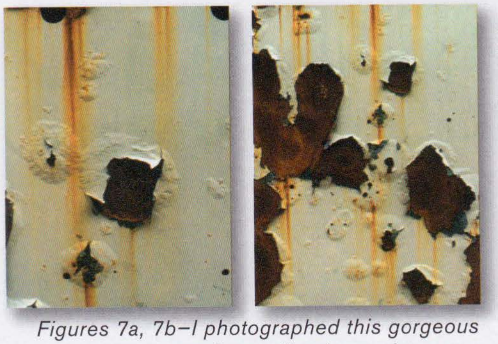


Figure 6c–After placing the harvested piece, I use the distort feature just enough to match the angle of the shingles and underlying plywood.



Figures 7a, 7b–I photographed this gorgeous texture showing paint peeling from a large gas container.

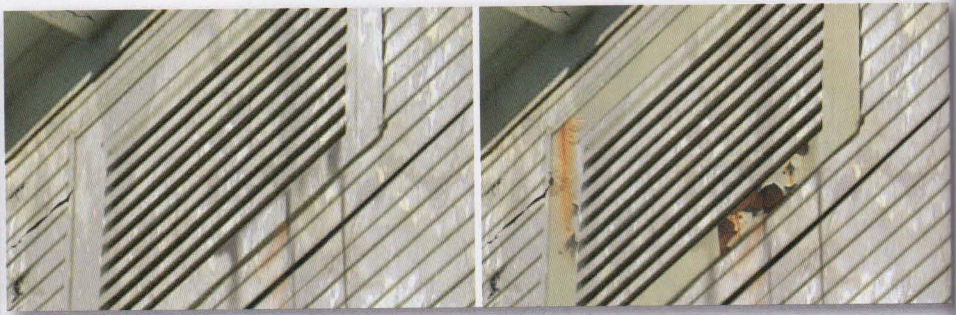


Figure 7c–After clipping pieces of the texture to the building, I sample that same yellowish color and paint over the rest of the vent.

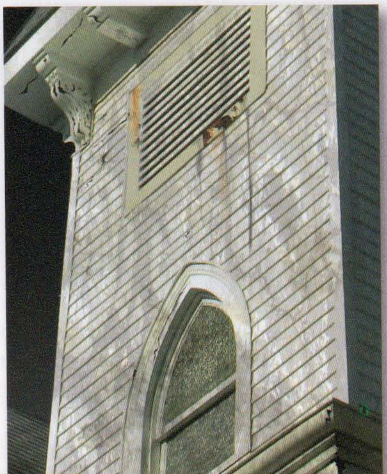


Figure 8a–Side of the church where I plan to show wood slipping and broken windows.

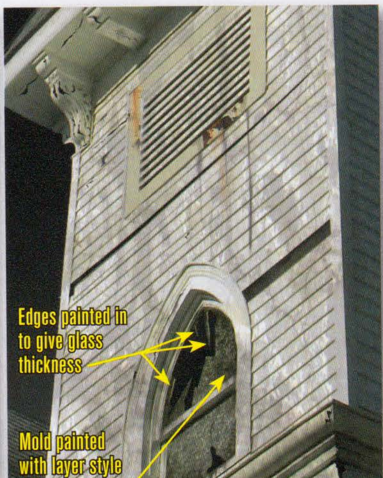
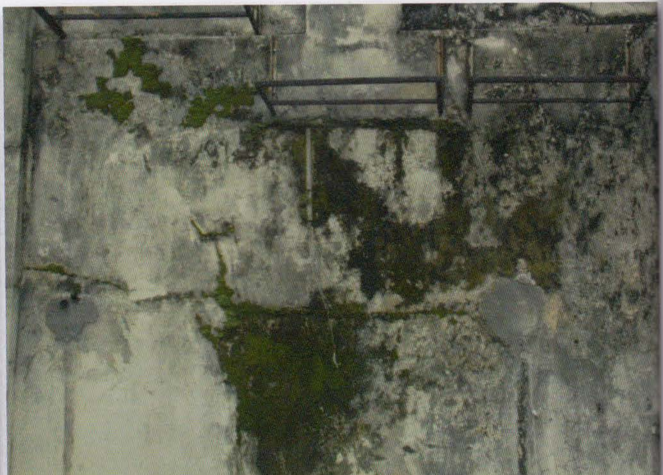

Edges painted in to give glass thickness
Mold painted with layer style effects

Figure 8b–Slipping wood and broken windows painted in along with mold on the windows.



Copyright ©2004 Mayang Adnin. See http://www.mayang.com/textures/

Figure 8c–Actual mold growing on a building.

plane, then erased away the edges of the damaged roof with a feathery eraser (**figures 6a-6c**).

For the metal vents near the top of the roof, I found some great texture of rusting and peeling paint from a similar surface and clipped it on there. As I stated before, I avoid stretching or sizing up textures, as that can cause blurring and degradation of the great texture you have. I do want to try and get the texture to feel like it is on the same angle as the vent here, so I used the distort feature to ease it into place, keeping a watchful eye out for any blurring. If you use a larger image than needed, the blurring can usually be kept to the far edges of the texture being imported and you can just erase that off (**figures 7a-7c**).

I'll also copy some of the wood slats from the original image onto their own layers and then tilt them and paint in shadow areas to give the impression of missing or falling wood.

When it comes to breaking the windows, I select paint in a very dark color–blackish brown or blackish green–to create the holes. Unless the room inside is lit, the holes will appear extremely dark. To give the impression of the glass having thickness, I'll paint a lighter gray edge here and there.

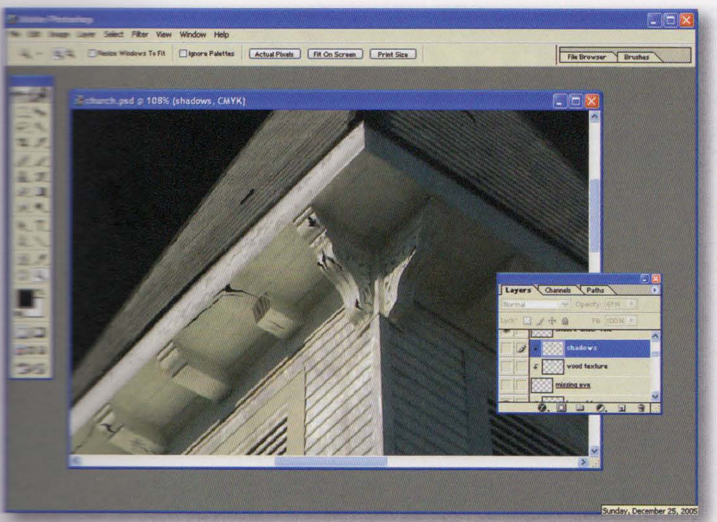

Adobe Photoshop
church.psd @ 108% (shadows, CMYK)
Layers
Channels
Paths
shadows
wood texture
missing eve
Sunday, December 25, 2005

Figure 9a–Top eave of the church where I plan to break a piece away.

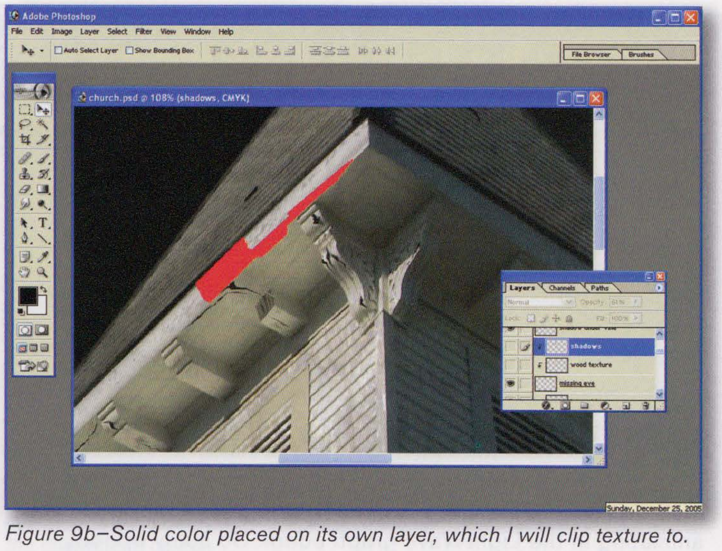

Adobe Photoshop
File Edit Image Layer Select Filter View Window Help
church.psd @ 108% (shadows, CMYK)
Layers
Channels
Paths
shadows
wood texture
missing eve
Sunday, December 25, 2005

Figure 9b–Solid color placed on its own layer, which I will clip texture to.

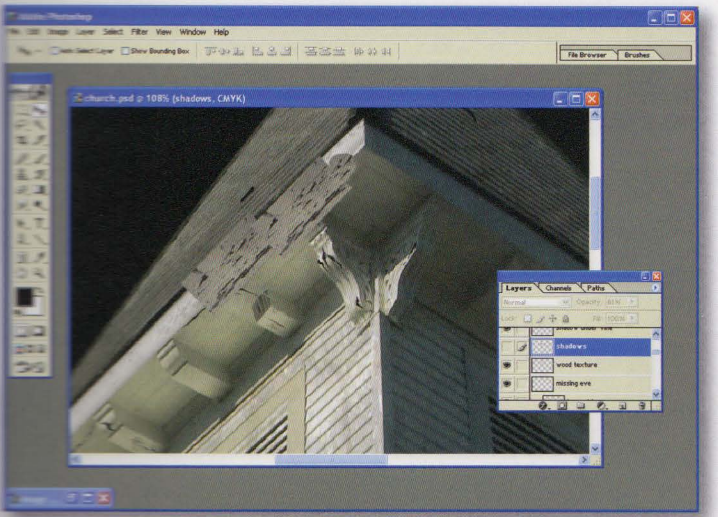

church.psd @ 108% (shadows, CMYK)
Layers
Channels
Paths
shadows
wood texture
missing eve

Figure 9c–Texture of rotten wood sits on a layer above.

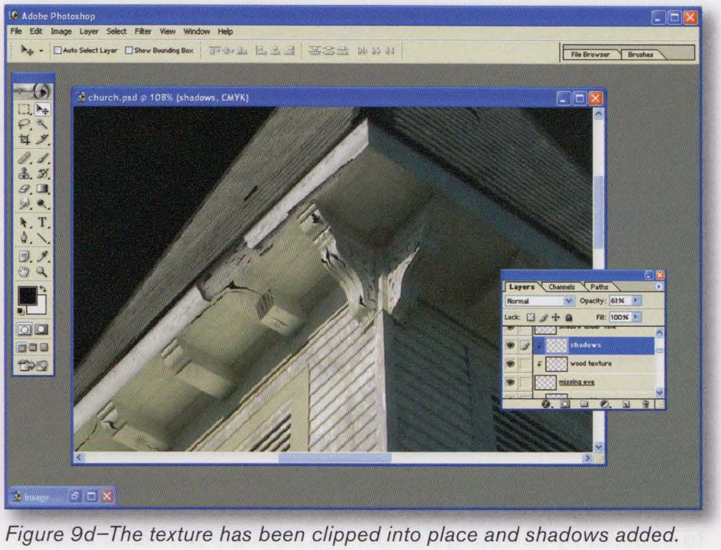

Adobe Photoshop
File Edit Image Layer Select Filter View Window Help
church.psd @ 108% (shadows, CMYK)
Layers
Channels
Paths
Normal
Opacity: 61%
Fill: 100%
shadows
wood texture
missing eve

Figure 9d–The texture has been clipped into place and shadows added.

To give the impression of molds, moss, and lichens growing on the building, you can find some texture and fix it to the building; or in this case, since the building is far enough away that I just need to give the impression of some 3D stuff growing on the surface, I'll paint and add a layer style effect. In this case, I used a bevel/emboss effect, then lowered the opacity on the layer itself so it blended with the existing image.

I do this on the wood, and also on the windows. Usually I steer clear of layer style effects in photorealistic work because the look can be so artificial, but the trick to making it work here is to have varied levels of opacity while you work. Start painting the mold in with various green colors and setting the opacity of the brush around 40% or so and use a texture brush. As the pigment builds up, closer to 100%, the more solid the bevel/emboss appears. Where you leave the paint more translucent, the effect is lessened and you end up creating sheets of mold and moss with a varied looking thickness to it, just like in nature (**figures 8a-8c**).

Along the top eave of the building, I gave the impression of a an entire piece of wood having fallen away, by painting in the contour of what the hole would look like, then clipping a piece of wood texture to it and adjusting that texture with levels and hue/saturation to make it look like it belongs on this building. To create the shadows, I used 100% black and then backed off on the opacity to blend with the texture below. When you are adding shadows to an image, take your visual cues from the other shadows that already exist in the image, and match the darkness to those (**figures 9a-9d**).

The final touches included adding a rambling vine growing up the side of the building and just a wisp of fog in the foreground (see title image on page 32). These ideas should help you get started with wearing out structures. See what other great effects you can come up with.

Briar Lee Mitchell *is a full time instructor at the Art Institute of Los Angeles, located in Santa Monica, where she teaches courses in visual effects and animation (www.aila.artinstitutes.edu). She has 30 years experience as an illustrator and a Master's Degree in Medical and Biological Illustration from the medical school, UCSF, in San Francisco. She has worked for The Smithsonian, Lucasfilm Ltd., Warner Bros., Johns Hopkins, Harvard Medical Institutions, Universal Studios, and Disney creating art for print, animation, live action, and gaming. Briar co-owns the gaming/animation company Star Mountain Studios with her business partner Richard Sternberg (www.starmountainstudios.com). You can view her animation/gaming work at www.paisleyshark.com and medical/forensic examples at www.medicalart-briarlee.com. Contact her at briarlee@aol.com.*

TransMotion Utilities Basics

By Aaron Kent
3D Animator
Brooklyn, New York

After Effects is a great tool for 2D animation. LightWave is an awesome full-featured 3D package. Imagine if you could combine the power of the two. Besides being the most powerful person in the galaxy, you might also stand to make a buck or two. That's not to say that I don't like animating in the LightWave graph editor. So many options, such smooth F-Curves... However, I find animating a large number of complex objects much easier to visualize and much quicker to preview when I'm in After Effects. Recently I found a plug-in package called TransMotion Utilities that is so much bang for the buck I figured I'd do a tutorial on it. For this tutorial let's animate an *HDRI 3D* logo fly-on. Laugh it up–but remember: If you work in broadcast, you'll be doing a lot of it, and they'll be paying you. Although this is fairly simple, the concepts can be applied to much more complex animations. Also, I suppose you could use the knowledge gleaned from this tutorial to animate "THE COOLEST 3D OBJECT YET CREATED" if you so chose.

STEP 1

Let's create our logo outlines in Illustrator. We'll save out one version to animate in After Effects, and another version as an EPS to build into 3D type inside Modeler. **Figure 1** shows our type in Illustrator. I traced the *HDRI 3D* logo so it's already outlined. If you're using the text tool, first, change your logo to outlines by selecting the text and from the file menu select *Type > Create Outlines*. Save this version as an Illustrator EPS. Next, let's take the same file and break each character (or logo element) out into its own layer, while also retaining a layer with the entire logo on it. **Figure 2** shows my layer palette after I've done so. Note: Name your layers–you'll thank me later when you have too many of them and you're trying to figure out which element "leftside logo thing copy3" refers to inside After Effects.

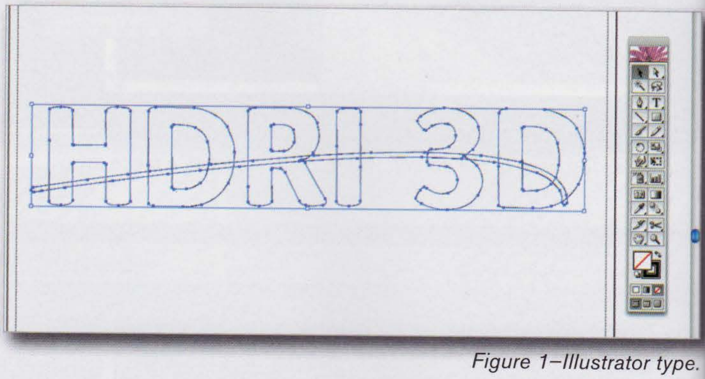


Figure 1–Illustrator type.

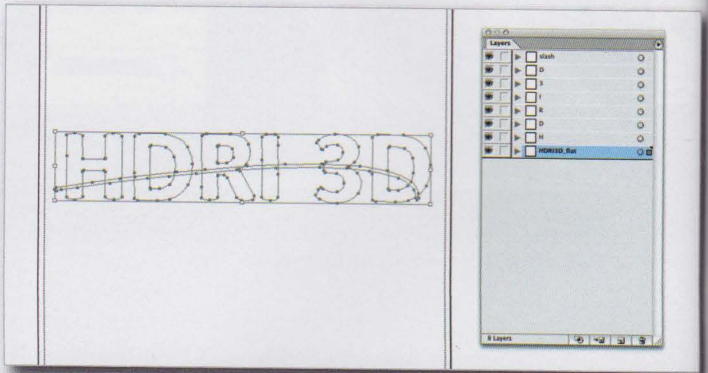
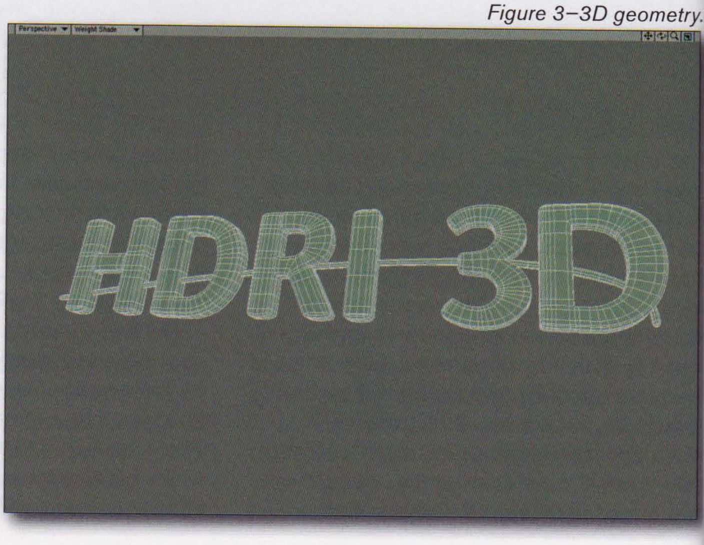


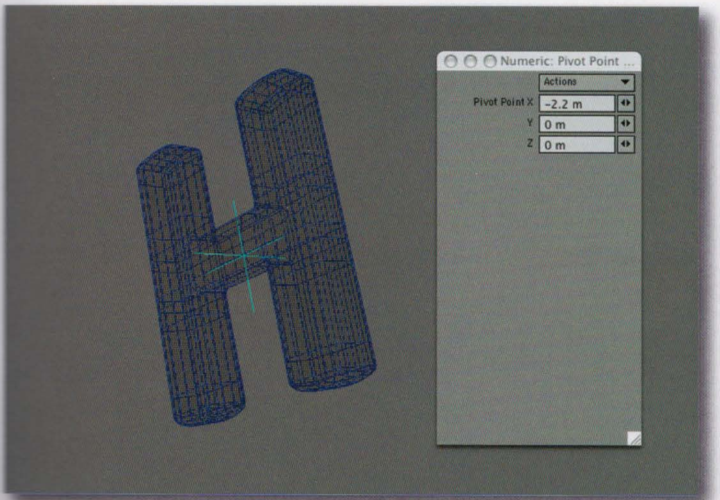
Figure 2–Name and label things appropriately! Good project management takes a back seat to nothing except good hygiene.

STEP 2

Model your text in LightWave Modeler. We won't touch on modeling in this tutorial, but now would be a good time to import your outlines into Modeler and start beveling them. From the file menu select *Import > EPSF Loader.* In the options dialog box, select the EPS file we just made and start modeling. **Figure 3** shows my logo all ready to go. If you're going to animate each character separately in After Effects, then the related character should be on its own layer inside LightWave. Also, it's important to have each layer's pivot point at the center of each object's geometry. The web offers a couple free plug-in options like Pivot Commander that accomplish this task. I have been known on occasion to just "eye it" as well. If this makes you cringe, just go to **www.flay.com** and run a search for "pivot" and you'll get what you need. Once I've centered all the characters' pivots I set the z value of all the characters to the same value. This will ensure that all the letters stay in line once I apply the After Effects animation to them inside Layout. To do this, I select the z value of one character's

Figure 3–3D geometry.





Numeric: Pivot Point ...

Pivot Point X: -2.2 m

Y: 0 m

Z: 0 m

Figure 4–The numeric panel activated for the pivot tool.

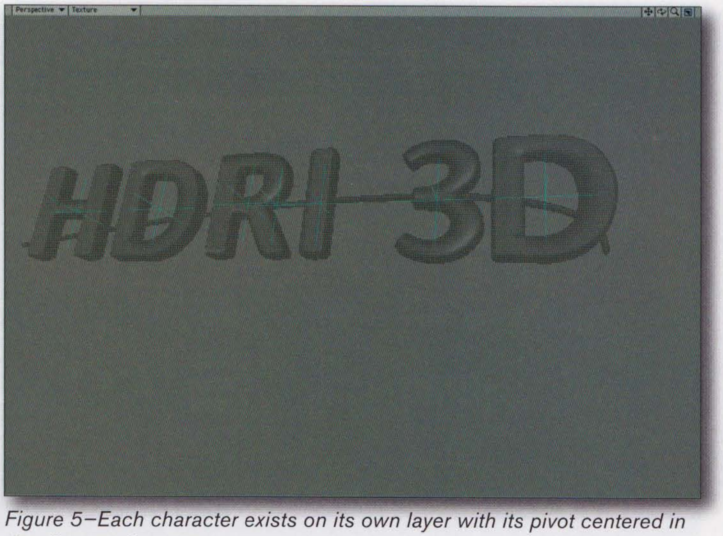
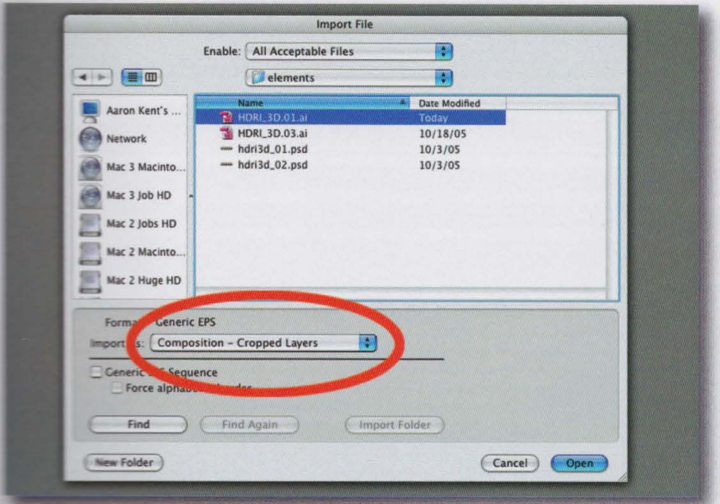

Perspective
Texture
HDRI 3D

Figure 5–Each character exists on its own layer with its pivot centered in the character's center.



Import File

Enable: All Acceptable Files

elements

Name	Date Modified
HDRI_3D.01.ai	Today
HDRI_3D.03.ai	10/18/05
hdri3d_01.psd	10/3/05
hdri3d_02.psd	10/3/05

Format: Generic EPS

Import As: Composition – Cropped Layers

Aaron Kent's ...
Network
Mac 3 Macinto...
Mac 3 Job HD
Mac 2 Jobs HD
Mac 2 Macinto...
Mac 2 Huge HD
Generic EPS Sequence
Force alphabetical order
Find
Find Again
Import Folder
New Folder
Cancel
Open

Figure 6–The After Effects Import dialog box. I've circled the "Import as:" selector. We want our Illustrator file to be imported with each layer cropped.

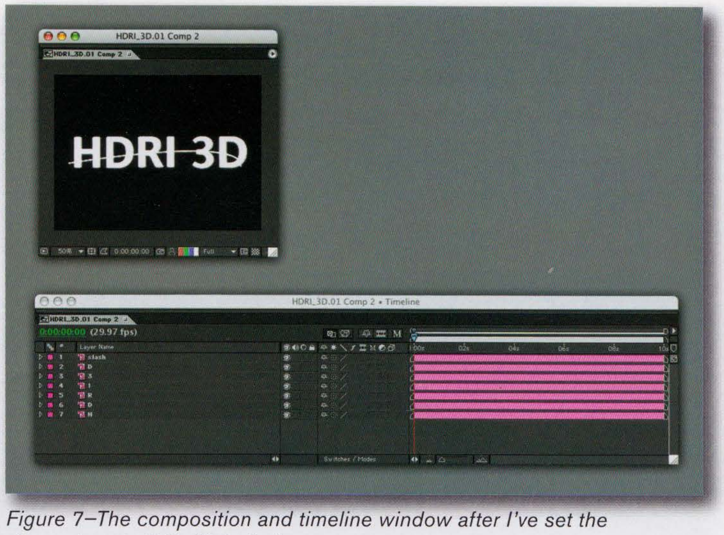

HDRI_3D.01 Comp 2
HDRI 3D
HDRI_3D.01 Comp 2 • Timeline
0:00:00:00 (29.97 fps)
slash
D
3
I
R
D
H

Figure 7–The composition and timeline window after I've set the dimensions to 648x486 pixels.

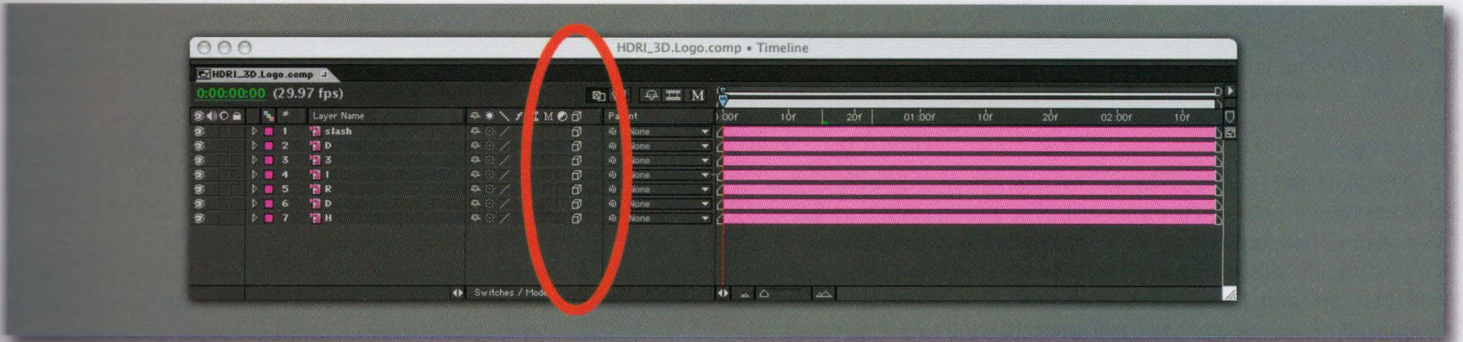
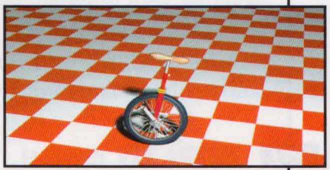

HDRI_3D.Logo.comp • Timeline
0:00:00:00 (29.97 fps)
Layer Name
1 slash
2 D
3 3
4 I
5 R
6 D
7 H
Parent
None
Switches / Modes

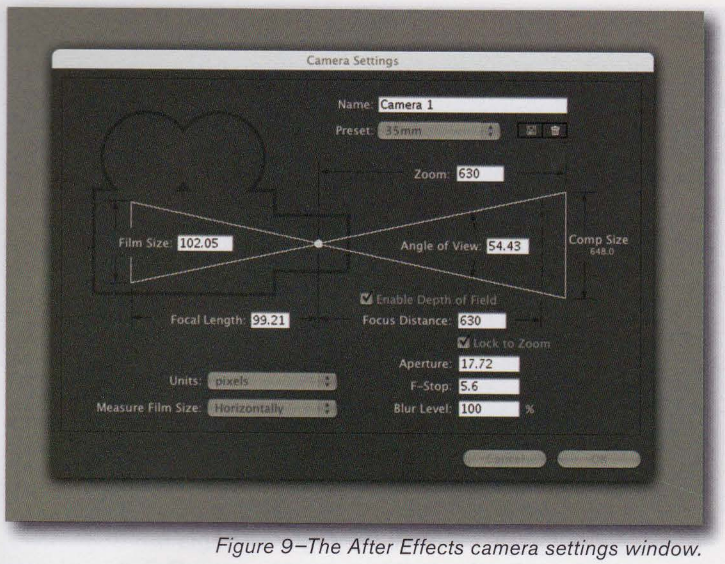
Figure 8–Enable those 3D layers.

pivot point and then apply it to each of the other characters. If you select the pivot tool and press the n button to see the numeric panel, you can copy and paste the values. **Figure 4** shows the numeric panel for a character's pivot location. **Figure 5** shows my model with the pivot points all visible.

STEP 3

Import your layered Illustrator file into After Effects and start animating. **Figure 6** shows the After Effects import dialog box. Let's import the Illustrator file using the cropped layered option. Importing as cropped layers centers the anchor point into the center of each Illustrator layer, which should match the pivots of our 3D characters closely enough. **Figure 7** shows my After Effects comp window with the Illustrator layers. It's also important to decide on the size of my composition. I'll be working at a resolution of 648x486 pixels. In LightWave I'll make sure to match my camera settings to this. The next step is to enable 3D space for each layer. This is activated by clicking the small cube icon on each layer in the sequence window (**figure 8**). Now I'll be able to move my text around in the X, Y, and Z axes. Finally, I'll add a camera in After Effects. Do this by selecting *Layer > New > Camera*.





Name: Camera 1
Preset: 35mm
Zoom: 630
Film Size: 102.05
Angle of View: 54.43
☑ Enable Depth of Field
Focal Length: 99.21
Focus Distance: 630
☑ Lock to Zoom
Aperture: 17.72
Units: pixels
F-Stop: 5.6
Measure Film Size: Horizontally
Blur Level: 100 %

Figure 9–The After Effects camera settings window.

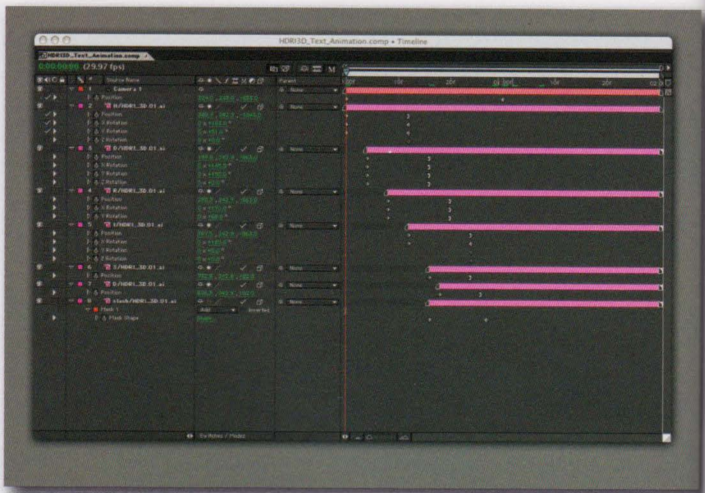
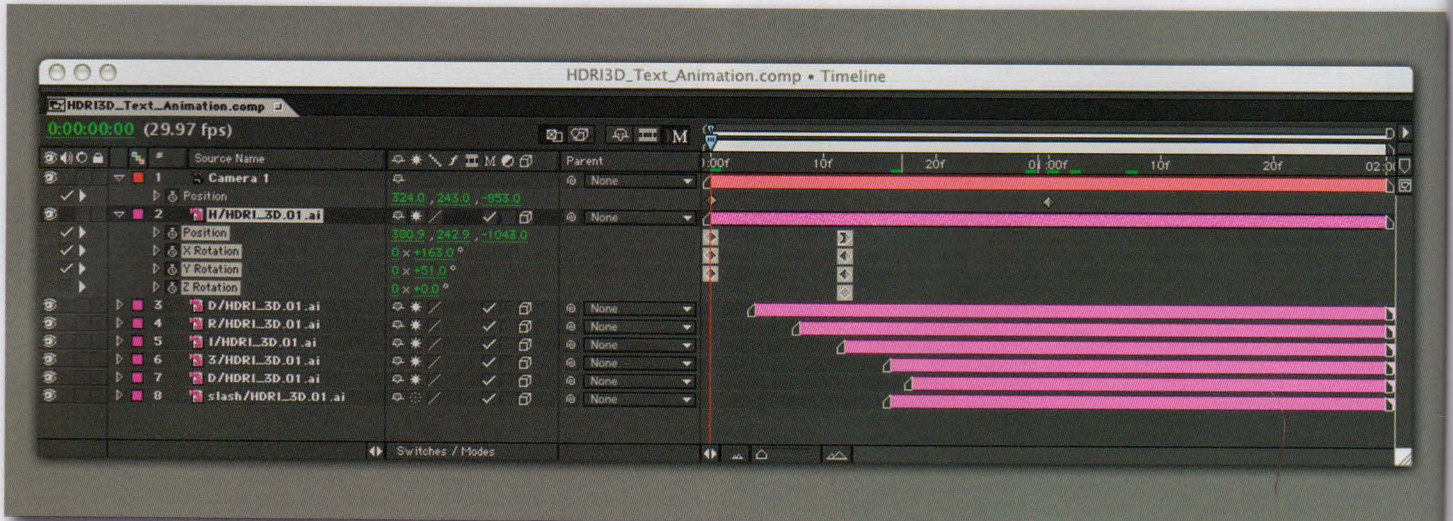


Figure 10–My After Effects timeline after I've fine-tuned my animation.



1 Camera 1 — None
Position 324.0 , 243.0 , -853.0
2 H/HDRI_3D.01.ai — None
Position 380.9 , 242.9 , -1043.0
X Rotation 0 x +163.0 °
Y Rotation 0 x +51.0 °
Z Rotation 0 x +0.0 °
3 D/HDRI_3D.01.ai — None
4 R/HDRI_3D.01.ai — None
5 I/HDRI_3D.01.ai — None
6 3/HDRI_3D.01.ai — None
7 D/HDRI_3D.01.ai — None
8 slash/HDRI_3D.01.ai — None

Figure 11–Dragging in the timeline window like a marquee tool allows you to select multiple keyframes in After Effects.

Figure 9 shows the camera options box. Leave the default settings and click OK to add the camera to your composition.

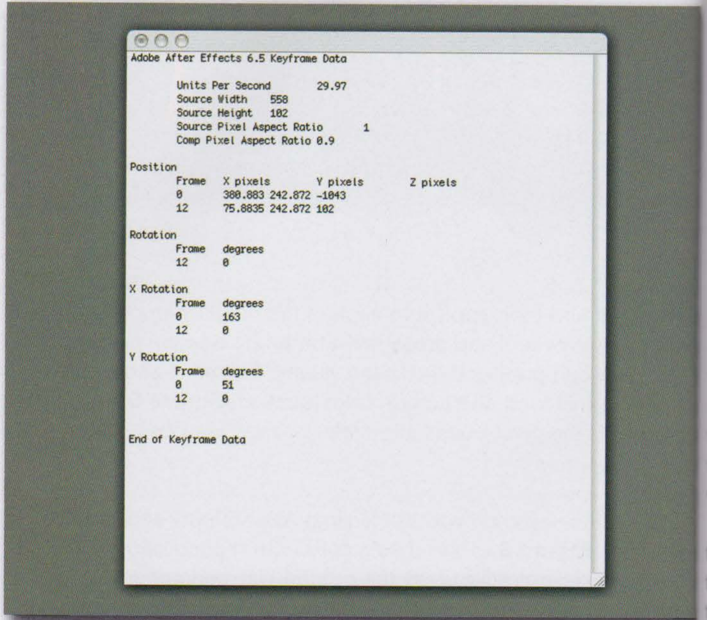
STEP 4

Time to animate in After Effects. This is where After Effects shines. RAM previewing lets you see your animation a lot quicker than the LightWave preview player and lets you choose from a number of different resolutions to see it in. I usually work and RAM preview at half resolution and then do a quick full resolution test render while animating text, logos, and the like. **Figure 10** shows my timeline after I've fine-tuned my animation and it's ready for export into LightWave.

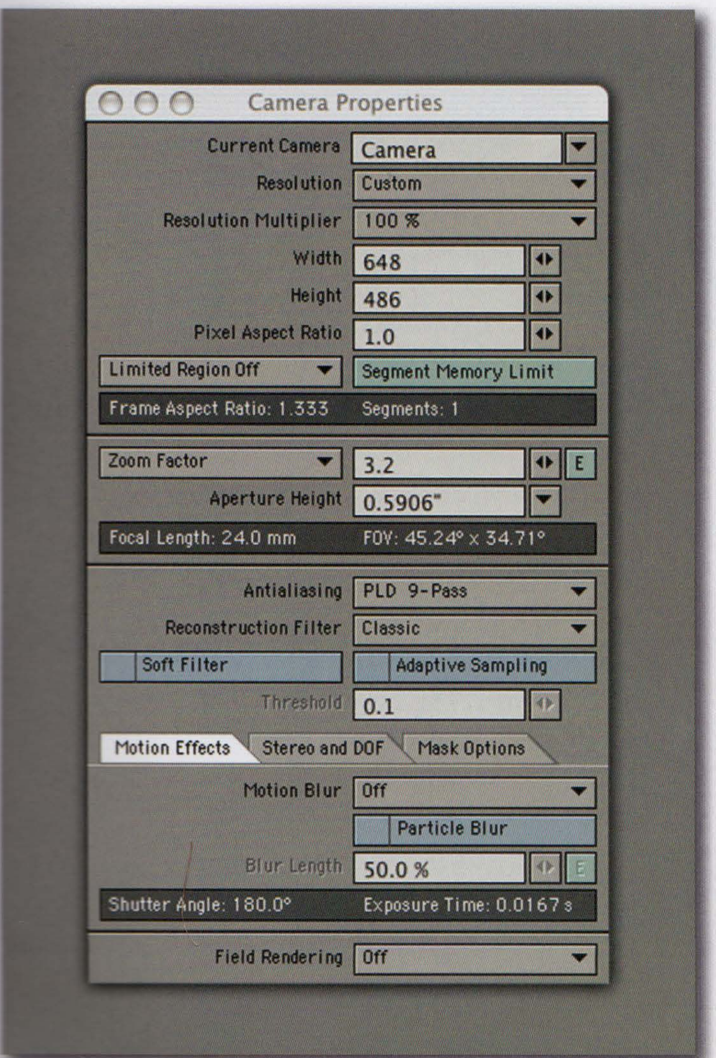
STEP 5

The next step is to copy our After Effects animation data into a format that the TransMotion Utilities can read. This is a simple process. For each layer in the After Effects timeline I select the layer, hit the "u" key to show all the keyframed channels, then select them and copy them. **Figure 11** shows the "H" layer keyframes selected and ready for copying. Then we open simple text or any other text editor and create a new file. Make sure that whatever text application you use, your preferences are set to create "plain text files." Then paste the information you copied from After Effects into the file. **Figure 12** shows the plain text file with the information pasted into it. After going through each layer (including the camera) in my

Figure 12–The selected keyframes translate to this when you paste them into a simple text document.



```
Adobe After Effects 6.5 Keyframe Data

	Units Per Second	29.97
	Source Width	558
	Source Height	102
	Source Pixel Aspect Ratio	1
	Comp Pixel Aspect Ratio 0.9

Position
	Frame	X pixels	Y pixels	Z pixels
	0	380.883 242.872 -1043
	12	75.8835 242.872 102

Rotation
	Frame	degrees
	12	0

X Rotation
	Frame	degrees
	0	163
	12	0

Y Rotation
	Frame	degrees
	0	51
	12	0


End of Keyframe Data
```



Camera Properties

Current Camera: Camera

Resolution: Custom

Resolution Multiplier: 100 %

Width: 648

Height: 486

Pixel Aspect Ratio: 1.0

Limited Region Off

Segment Memory Limit

Frame Aspect Ratio: 1.333 Segments: 1

Zoom Factor: 3.2

Aperture Height: 0.5906"

Focal Length: 24.0 mm FOV: 45.24° x 34.71°

Antialiasing: PLD 9-Pass

Reconstruction Filter: Classic

Soft Filter

Adaptive Sampling

Threshold: 0.1

Motion Effects | Stereo and DOF | Mask Options

Motion Blur: Off

Particle Blur

Blur Length: 50.0 %

Shutter Angle: 180.0° Exposure Time: 0.0167 s

Field Rendering: Off

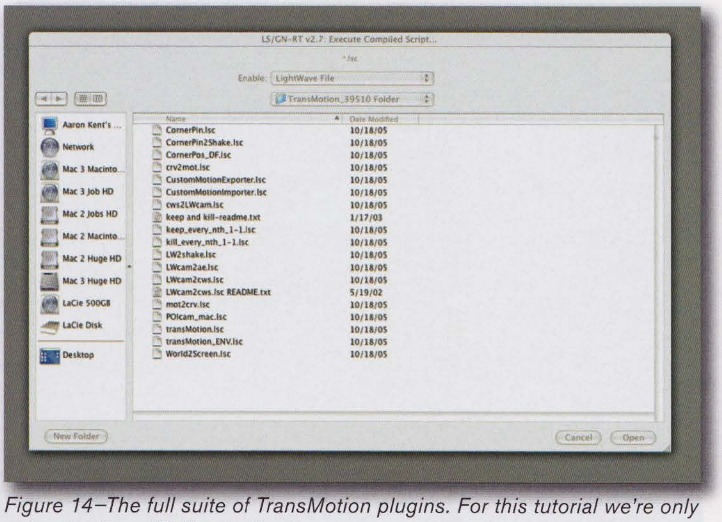
Figure 13–The LightWave 3D camera properties window. We won't get into focal length or zoom in this tutorial–we'll just match the dimensions of our composition settings from After Effects.

sequence and creating a simple text document with its keyframe information copied in, I'm ready to move on to the next step.

STEP 6

Off we go to LightWave. Now it's time to load our 3D text and apply our After Effects animation. First off, I create a new scene in LightWave and set up my composition settings. Our After Effects comp was 648x486 pixels, so I'll go to camera settings and enter those values. **Figure 13** shows my camera settings panel in LightWave. Next, let's import our modeled geometry. Once that's done we can start applying the animation. The TransMotion Utilities are a collection of LScripts. A demo version is available to try them out. They can be downloaded from **http://www.ats-3d.com/transmot/**. The demo limits the plug-ins' capabilities to a certain numbers of keyframes or buffers, but you should be able to get a feel for them regardless. There is a downloadable manual, which is clear and precise, and a number of tutorials on the Web site that are all easy to follow. Also, once you purchase the plug-ins, they are keyed to your LightWave dongle so they'll work when and where you need them whether it's at work, home, or on your laptop.

OK, that said, we have a little grunt work to do. I have to apply the TransMotion plug-in to each layer of my geometry. I select the "H" layer in LightWave. Then I click on the Utilities tab at the top of LightWave layout and select Lscript R/T from the tool menu. This will open up a browser. Navigate to wherever you have the plug-ins installed and select the main TransMotion Utilities plug-in. **Figure 14** shows the entire collection of TransMotion plug-ins. Select transMotion.lsc and click the OK button. The TransMotion dialog box should now open. **Figure 15** shows the dialog box in all its glory. The manual contains a good description of what all these options do. For this tutorial's purpose, we have to do the following:



LS/GN-RT v2.7: Execute Compiled Script...

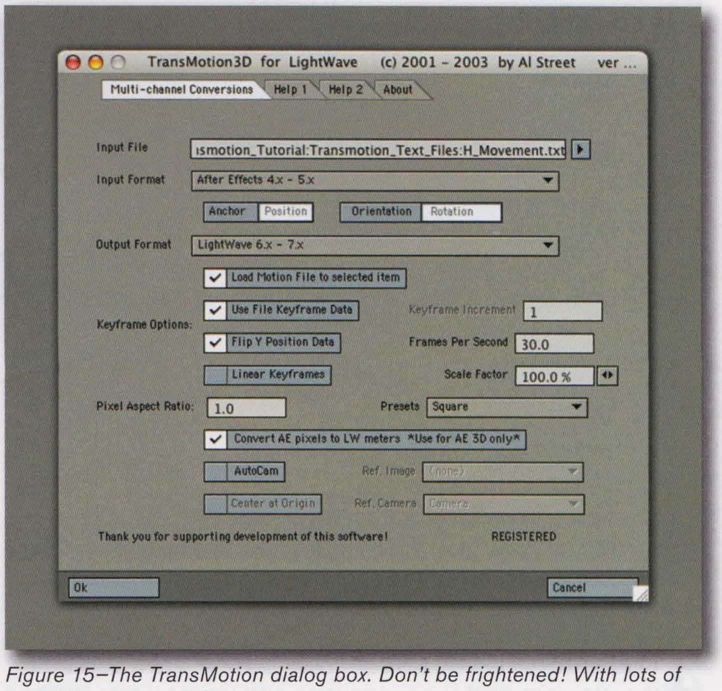
*.lsc

Enable: LightWave File

TransMotion_39510 Folder

Name	Date Modified
CornerPin.lsc	10/18/05
CornerPin2Shake.lsc	10/18/05
CornerPos_DF.lsc	10/18/05
crv2mot.lsc	10/18/05
CustomMotionExporter.lsc	10/18/05
CustomMotionImporter.lsc	10/18/05
cws2LWcam.lsc	10/18/05
keep and kill-readme.txt	1/17/03
keep_every_nth_1-1.lsc	10/18/05
kill_every_nth_1-1.lsc	10/18/05
LW2shake.lsc	10/18/05
LWcam2ae.lsc	10/18/05
LWcam2cws.lsc	10/18/05
LWcam2cws.lsc README.txt	5/19/02
mot2crv.lsc	10/18/05
POIcam_mac.lsc	10/18/05
transMotion.lsc	10/18/05
transMotion_ENV.lsc	10/18/05
World2Screen.lsc	10/18/05

Figure 14–The full suite of TransMotion plugins. For this tutorial we're only using the "transMotion.lsc."



TransMotion3D for LightWave (c) 2001 - 2003 by Al Street ver ...

Multi-channel Conversions | Help 1 | Help 2 | About

Input File: ısmotion_Tutorial:Transmotion_Text_Files:H_Movement.txt

Input Format: After Effects 4.x - 5.x

Anchor | Position   Orientation | Rotation

Output Format: LightWave 6.x - 7.x

☑ Load Motion File to selected item

Keyframe Options:

☑ Use File Keyframe Data   Keyframe Increment: 1

☑ Flip Y Position Data   Frames Per Second: 30.0

☐ Linear Keyframes   Scale Factor: 100.0 %

Pixel Aspect Ratio: 1.0   Presets: Square

☑ Convert AE pixels to LW meters *Use for AE 3D only*

☐ AutoCam   Ref. Image: (none)

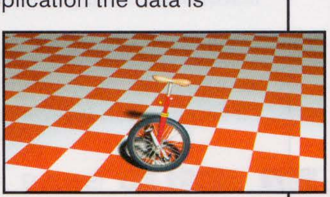
☐ Center at Origin   Ref. Camera: Camera

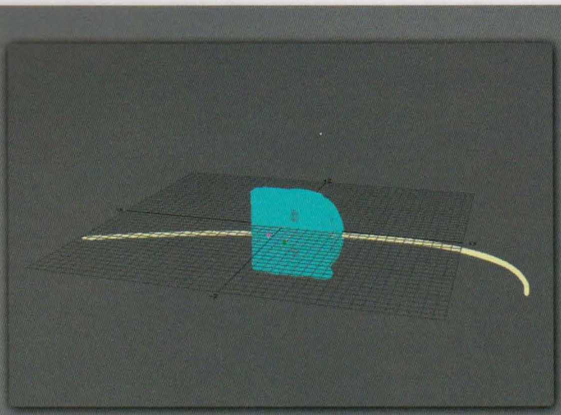
Thank you for supporting development of this software!   REGISTERED

Figure 15–The TransMotion dialog box. Don't be frightened! With lots of options comes lots of power.

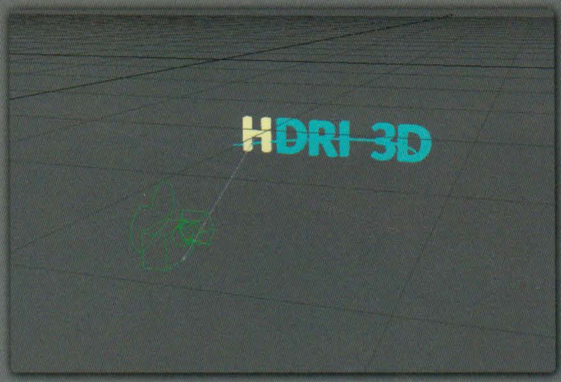
1 Input File: Here we select one of the simple text files we created by copying data from After Effects. Since the "H" geometry is selected, we should navigate to the "h" simple text file we created.

2 Input Format: Tells the plug-in what application the data is coming from. In our case, select the After Effects option.





before we correct the objects scale



after we correct the objects scale

Figure 16–The animation is correct, but until I adjust the scale, things can look quite odd.

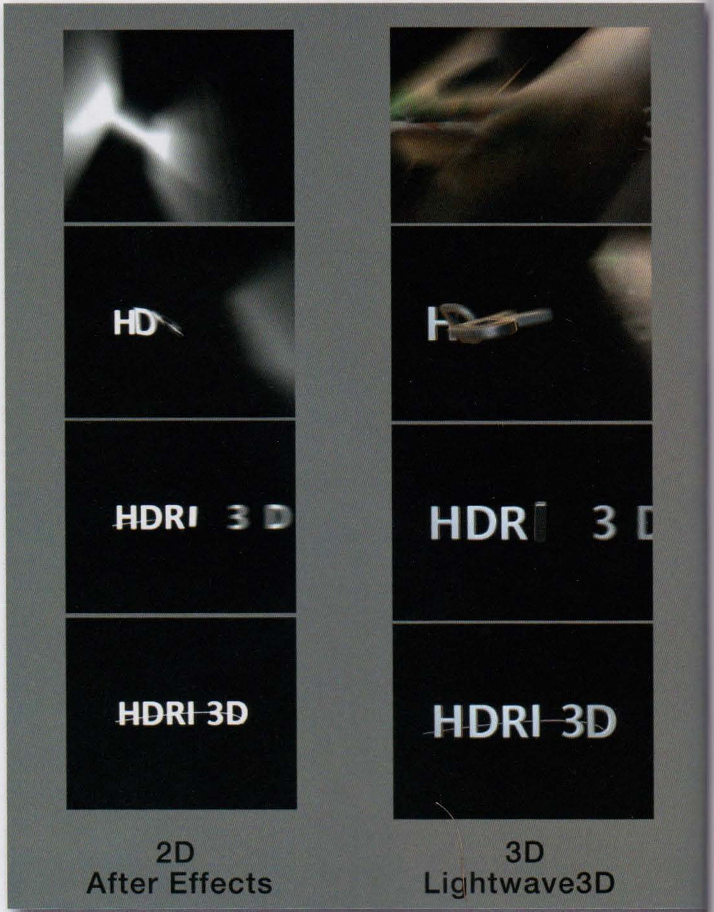


Figure 17–Before and after. What we start with in 2D starts to really fill out in 3D.

3 We select position and rotation instead of anchor and orientation. The plug-in allows for any of these attributes. We could have animated the anchor and orientation of our text in After Effects had we decided on that route.

4 Output Format: This tells the plug-in what we're converting the data into. In this case, motion and rotation data inside LightWave 3D.

5 Let's check "Load Motion File to selected item." This one is pretty self explanatory.

6 Let's check "Use File Keyframe Data." We want the animation for each layer to respect the start and end frames we decided on in After Effects. If we didn't check this box, I imagine everything would start moving right at frame 0 in LightWave.

7 Let's check "Flip Y Position Data." Apparently After Effects and LightWave 3D have opposite Y values in their coordinate systems. This corrects for that.

8 Let's check "Convert AE pixels to LW meters." I don't know exactly what this does, but the manual says to use it if we are working in After Effects 3D space.

Once we've set all these values, click OK and the animation should now be applied to the "H" geometry. From here on I'll select each layer in LightWave 3D and apply the correct motion file to it with the TransMotion plug-in. Finally, I'll select the camera and use the plug-in to apply the camera data.

STEP 7

My animation is now transferred, but I have one issue left to address. I haven't figured out how to maintain a relationship as to the scale of the geometry and flat text between After Effects and LightWave 3D. My animation is correct, but my geometry is scaled so large inside LightWave that all the text overlaps and doesn't appear to "fly in" to the camera's view. I'll fix this by adjusting the scale of the geometry in LightWave 3D after I've applied my animation. For the animation in this tutorial, I scaled each piece of geometry to .040 percent of its original size. **Figure 16** shows the logo's rest position in LightWave 3D with and without the change in scale.

Well, that's about it. **Figure 17** shows a frame of the After Effects text animation next to a frame of the LightWave final animation. I hope this tutorial helped a little bit and got you thinking of ways you can harness a really great (and ultra low cost) plug-in for LightWave 3D.

***Aaron Kent** is an After Effects and LightWave Artist working in New York City. During the day he works at a post house called Vizualsolutions (www.vizualsolutions.com). If he ever gets any of his personal 3D work done, you'll find it at www.shiftlessdreams.com. He runs in true 64-bit mode all the time, even when sleep deprived. He lives in Brooklyn.*

THE TWO FACES OF T-SPLINES

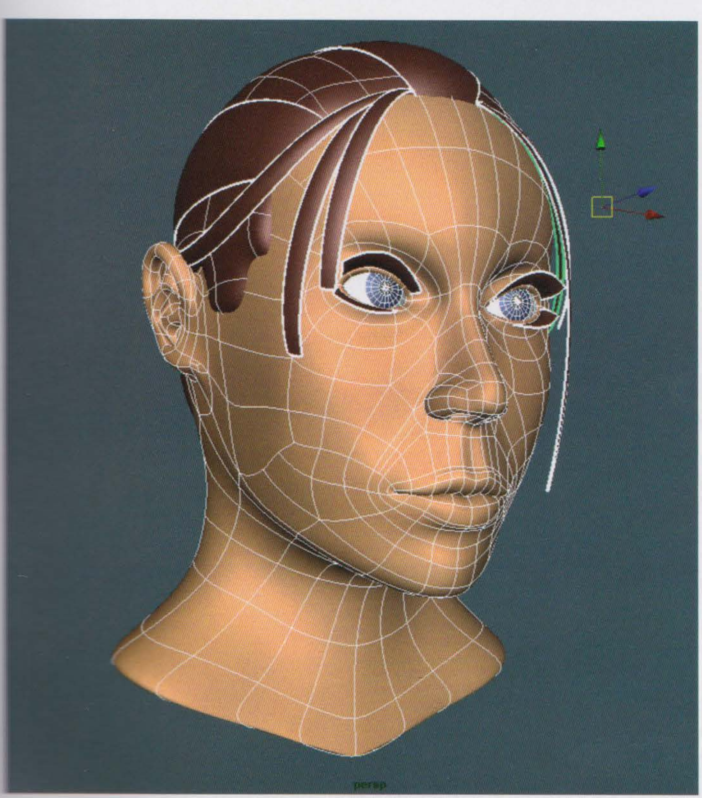


Figure 1—Version 1: Subd-eque mesh layout.

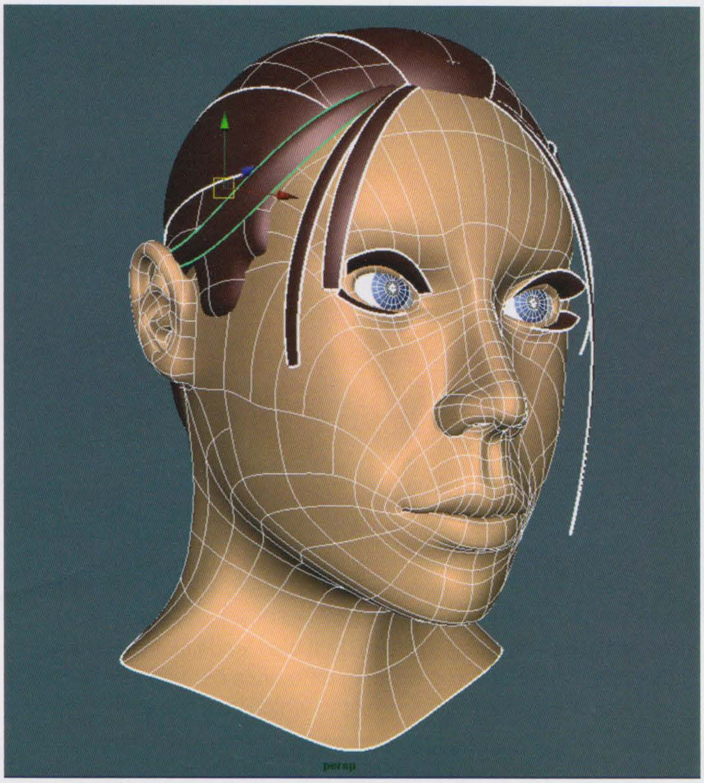


Figure 2—Version 2: NURBS-like mesh layout.

By Eric Allen
Modeling Lead
Draper, Utah

I was introduced to the powerful concept of T-Splines at SIGGRAPH '04. I found the technology to be based on one uncomplicated idea: NURBS surfaces do not need to be bound by the fact that resolution has to continue to the edge of the surface. There is much going on "under the hood" to achieve this, but to the artist, it is like working with the freedom of subds while enjoying the benefits of NURBS. If only NURBS could have started out like this! Time spent seaming NURBS surfaces could have been spent on actual modeling. In this tutorial of modeling two female heads, I share what I have learned about how to use the T-Splines surface type efficiently. I used the T-Splines Maya Plugin with Maya 6.0. The plugin is also available for 6.5 and 7.

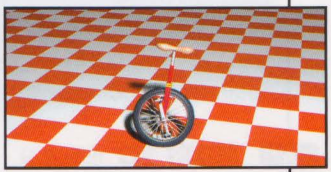
CHOOSING YOUR T-SPLINES WORKFLOW

One of the benefits of T-Splines is that it gets along well with other geometry types. NURBS, subds, and polygons can all be converted to T-Splines, and T-Splines can be converted to NURBS and polygons. Given these options, I like to rough out my models in polygons, then convert them to T-Splines to add detail and remove unnecessary geometry. I choose this method because polygons have few constraints and allow for easy planning of future resolution. If you want to convert existing NURBS models to seamless T-Splines models, it is likewise not difficult. It takes a little getting used to and is sometimes tricky when working with multi-patch seams, but after a few minutes (aided by using the T-Splines tutorials on ***www.tsplines.com***) it is easy to figure out. If you are starting from scratch, I recommend using polys.

PREPARING YOUR POLYGON MODEL WITH T-SPLINES IN MIND

The optimal layout of your polygonal, pre-T-Splines model will depend on what rendering constraints you are working within and what you plan to use the model for. Normal consideration must also be taken for opposing geometry (for animation) and correct resolution detail in areas that will move. This is normal procedure for most modelers.

I have prepared two T-Splines models: a subd-esque mesh layout version (1) (**figure 1**) and a cleaner, more rectangular NURBS-like mesh layout version (2) (**figure 2**), each representing a different workflow. Many game and



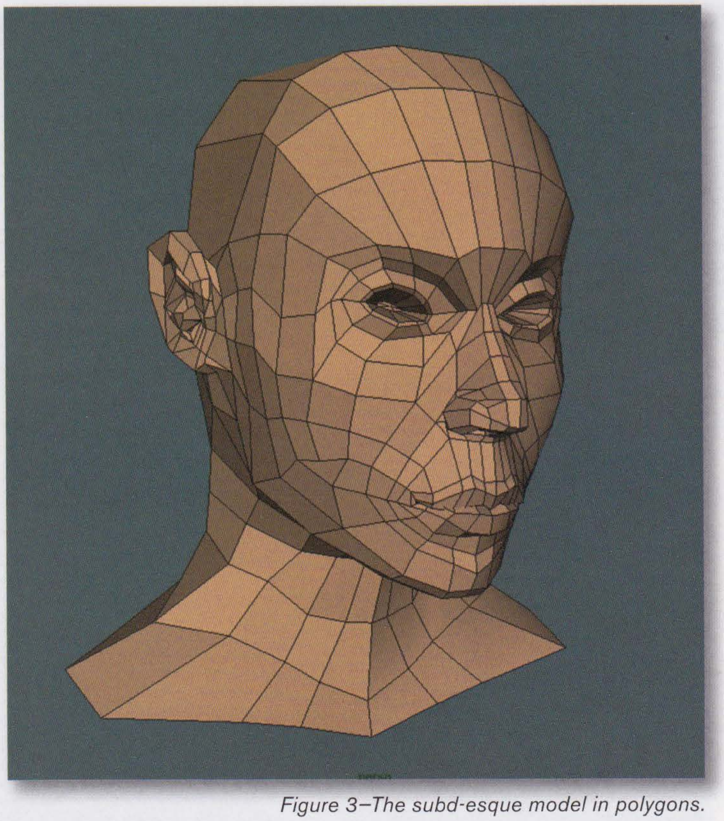


Figure 3–The subd-esque model in polygons.

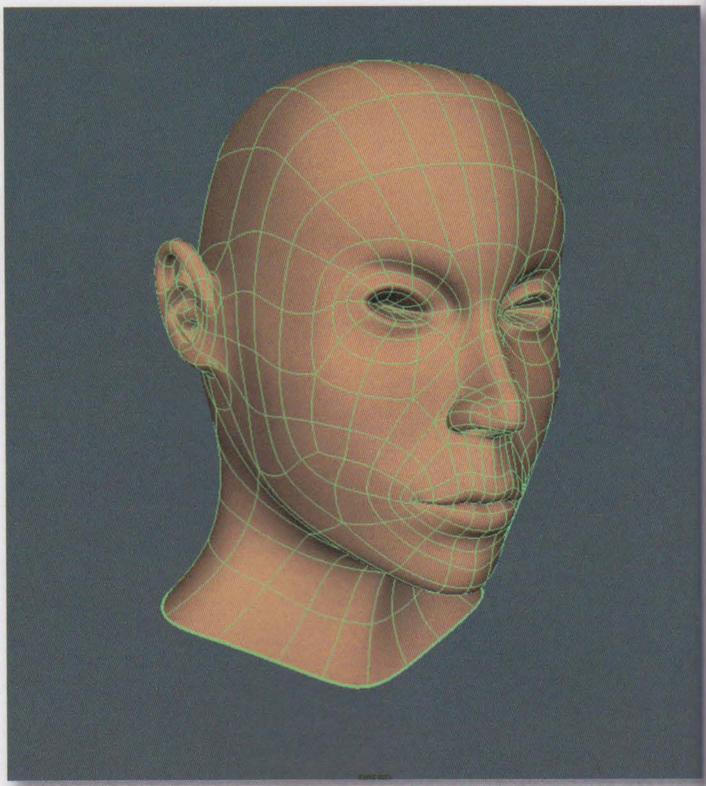


Figure 4–Converting the subd-esque model to T-Splines.

effects studios will prefer the ease and speed of the subd-esque version, while many high-end animation houses and architectural designers will likely require the exactness of version 2, which could more easily be outputted to NURBS. It also lends itself to cleaner, higher quality renders. The defining difference between these two workflows in T-Splines is the extent to which they use extraordinary points.

EXTRAORDINARY POINTS: TO USE OR NOT TO USE?

Extraordinary points are points in the model where more or less than 4 faces come together. In subdivision surfaces, they are sometimes called poles (less than 4 faces) or stars (more than 4). They often appear with edge loops around highly animated parts of the model, such as the mouth and eyes. These points are used when differing directions of polygonal geometry meet each other. It is good practice, however, to keep these points to a minimum, or hidden if you can. You'll notice that the subd version (1) features several prominent extraordinary points.

By contrast, I minimized the use of extraordinary points in creating the second version. This minimalist approach is important because if you convert your T-Splines to NURBS further down the pipeline, the resulting NURBS patches will be generated to have corners at extraordinary points. The more extraordinary points your model has, the more NURBS patches will be created. As the number of NURBS patches increases, there are more tangencies to deal with, which may be hard to control. To alleviate this, you will want to minimize the number of extraordinary points and compose your original poly model as a grid as much as possible. Planning beforehand will save time later. Depending on your workflow, you will be able to decide which method is best to use.

MODELING THE SUBD VERSION (1)

When preparing the polygonal layout for this version, I took a standard subd approach (**figure 3**). I used edge loops around areas for the purpose of a clean layout and future animation. This invariably led to extraordinary points. However, I limited the number of extraordinary points by using just the edge loops I needed for basic shaping. I postponed adding details until after I had converted to T-Splines, since this is one of the more powerful reasons for using this new surface type.

After converting to T-Splines (**figure 4**), I added detail with T-Junctions, which gave me the detail I needed with less geometry than polys. In polygonal and subd modeling, there

T-JUNCTIONS

One thing to be aware of: In T-Splines you can have T-Junctions as well as extraordinary points. T-Junctions are points that terminate a partial isoparm, and are much better than extraordinary points because they produce cleaner surfaces. If you have a choice between using a T-Junction or an extraordinary point, you should generally choose the T-Junction. Since three edges meet at a T-Junction, it is easy to mistake a T-Junction with an extraordinary point. To avoid confusion, you can color the extraordinary points differently from the T-Junctions by selecting *TSplines > Display Options > Extraordinary Points.*

CONVERTING THE POLYGON MODEL TO T-SPLINES

Regardless of the approach you are taking, once you have your polygonal model roughed out, converting it to T-Splines is quick: Simply click on the Polygon-to-T-Splines button or select *T-Splines > Convert from Polygons* from the pull-down menu.

needs to be a tapering out of high resolution geometry to drop resolution into low resolution mesh–details such as creases around the mouth would have required more geometry for tapering. The actual operation of inserting detail is very simple–simply select the edges you would like to increase the detail on, and select *TSplines > Insert Point.*

While my main goal in this version was looking for ways to increase the detail, I also cleaned up my model while I went along by deleting unnecessary edges in the cheek and forehead. This smoothed those parts of the model while also decreasing the amount of geometry. I deleted the large areas on the cheek mostly to demonstrate that you can take out as much or as little as needed.

Also, by using T-Junctions, I avoided N-gons (polys with 5+ points), which don't always shade as desired. In contrast, the T-Junctions are smoothed automatically for you. I was also able to easily increase the resolution as much as I needed within the span of a polygon for things such as the details in the ear and eye.

Those same basic techniques of inserting control points and extending T-Junctions in the ears allowed me to add detail to the rest of the model, as seen in **Figure 1**: the fold between nose and mouth, the epicanthal fold (crease of skin above the eye), philtrum (area under the nose), and the edge defining the upper lip. This process could also be used for additional detail, like age wrinkles, crow's feet, and other detail needed in realistic animation.

When adding detail in T-Splines (**figure 6**), the surface will not deform. When adding geometry to a subdivision surface without additional modification, unwanted deformation appears in the flow of the subd mesh. Similar T-Splines manipulations will compensate for the added resolution and will maintain the integrity of the shape and flow of the surface. This feature of T-Splines is very important in that, instead of fixing surface deformations and minute nuances created by adding resolution, time can now be spent on actual modeling.

There are a number of other additional tools available in T-Splines to assist in the process of adding detail, such as creasing, extrusion, and extending T-Junctions. Creasing is effective when you need a really sharp edge. The tool places partial isoparms next to your edge selection to tighten the area. In dealing with extrusions, I prefer to take care of those early before the conversion to T-Splines. I find that if I place T-Junctions wisely, I usually don't need to extend them. However, it is nice to have the tools there if you need them.

Another powerful tool is the merging tool. This tool seams two T-Splines surfaces together into one surface. It can be used when attaching hands to arms or a head to a torso. Merging is simple to use. You just select the edges and execute. You then get a single smoothed surface. You can also merge together three surfaces that meet at one point.

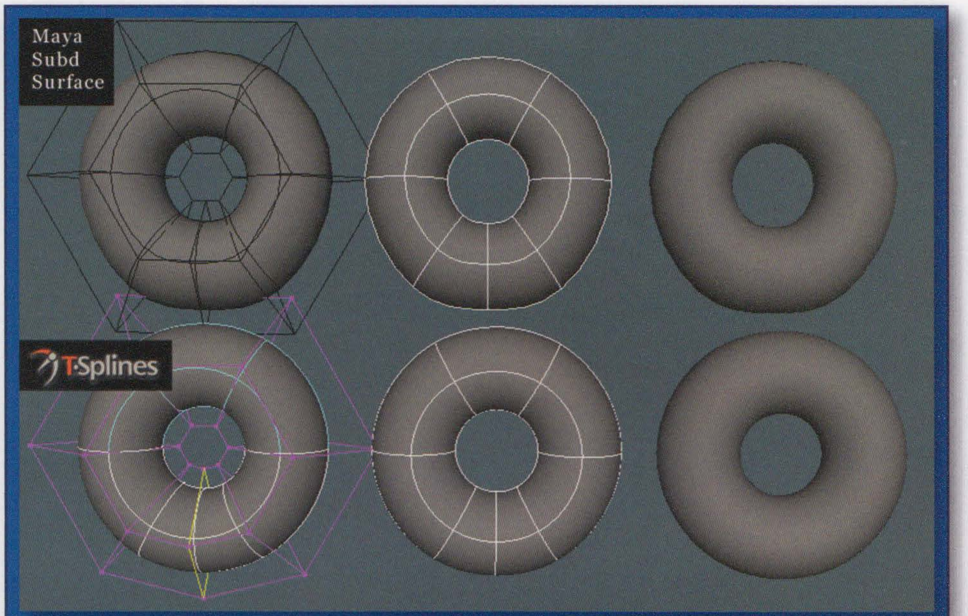
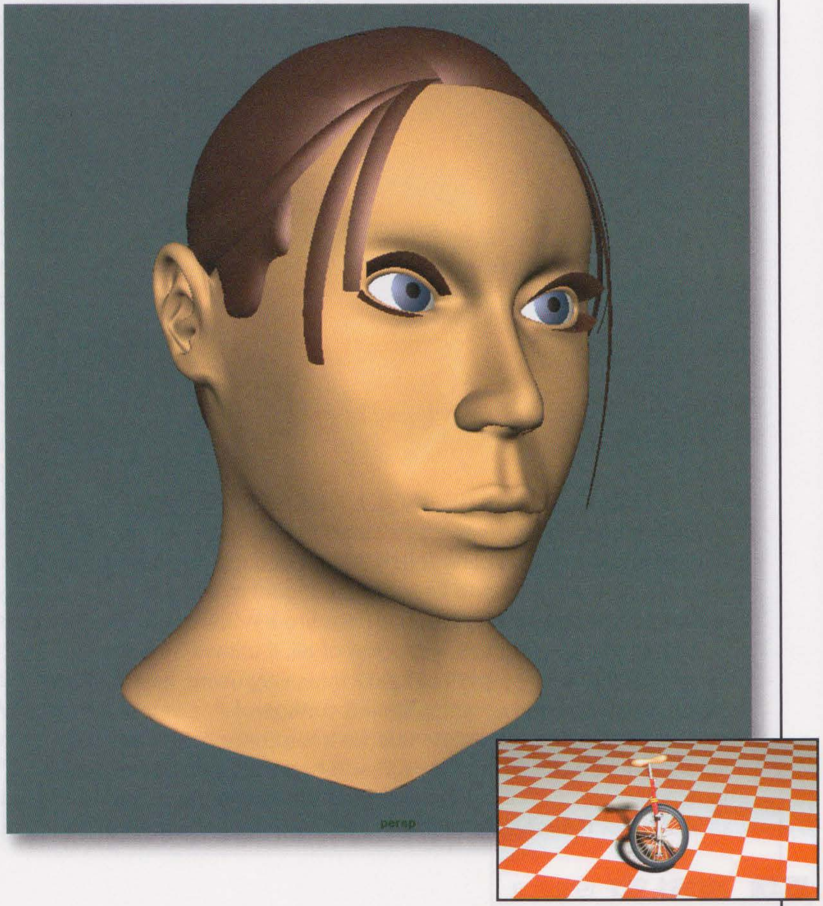


Figure 5–T-Splines keeps the integrity of the shape after inserting detail. Here, similar amounts of detail were inserted on both a subd and a T-Splines torus. Observe that while the subd torus deformed, the T-Splines torus kept the exact same shape.

COMPARING SUBD AND T-SPLINES RESOLUTION ADDITIONS

When a subdivision control mesh edge is split, there is a resulting unwanted deformation in the subdivision surface. When the same manipulation is applied to a T-Splines edge, the plugin compensates for this and maintains the exact shape of the original surface. **(figure 5)**

Figure 6–Subd-eque model ready for rendering and animation.



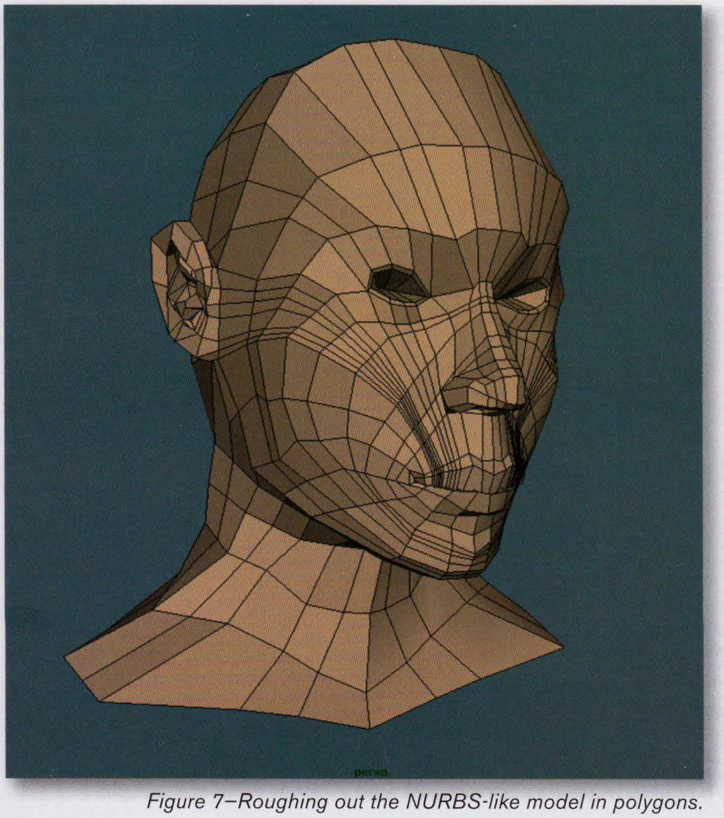


Figure 7—Roughing out the NURBS-like model in polygons.

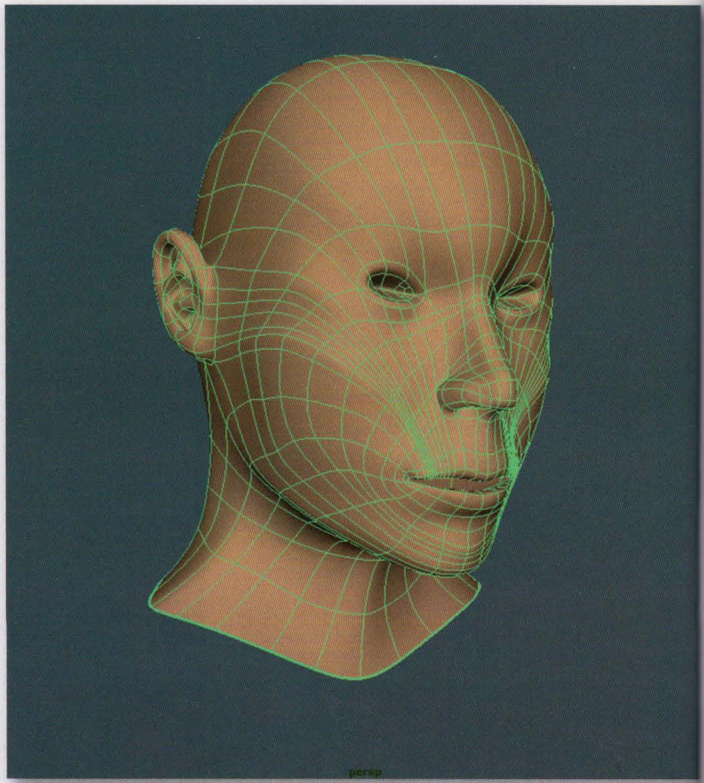


Figure 8—Converting the NURBS-like model to T-Splines.

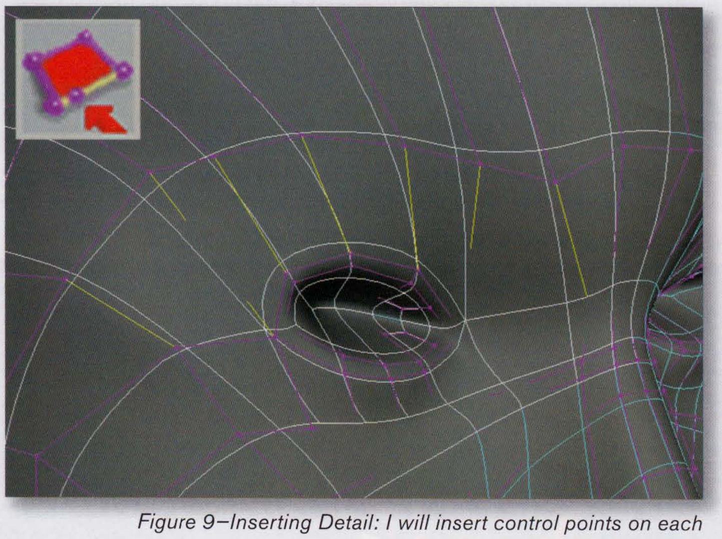


Figure 9—Inserting Detail: I will insert control points on each of the yellow edges.

This is a little trickier, but it is a very powerful and welcome option when you need it.

MODELING THE NURBS-ESQUE VERSION (2)

With this second approach, I decided to stay more within the paradigm of traditional spline (NURBS) modeling, minimizing the extraordinary points and keeping the model in more of a rectangular topology (**figure 7**). There are a number of advantages in modeling this way. First of all, it allowed me to define the detail of my model almost exclusively using T-Junctions instead of extraordinary points. The advantage of this is that the resulting T-Junctions are deleteable (extraordinary points are not deleteable in T-Splines) and they shade better than extraordinary points. This took additional setup time at the beginning when I was laying out the mesh, and there were more edge loops for the purpose of having resolution where I needed it. Second, this method allowed me the freedom of deleting areas in which I did not need the resolution, and since I was using a spline-based surface type, smoothing was fast.

Finally, extraordinary points can still be used, although you'll want to add them into areas where they can easily be hidden. I chose to use these in the nose and ear areas because of the needed direction changes. In NURBS, I might have had to add an additional extra surface for the ear model and then smooth it in at the border edges, or in the case of the nose, I would have had to used patches for detail and geometry flow. I didn't need to worry about any of these issues, thus saving time for more important things. It is still amazing to me that I can utilize this free type of mesh flow thinking in a NURBS-type surface. It was neat to model in polys while thinking about NURBS continuity. As I converted over to T-Splines, the planned resolution density and direction was there; thus, it was simple to add or take away detail wherever I needed.

There are two ways to obtain the necessary resolution: You can either add the detail beforehand and delete out the unnecessary geometry, or you can lay out a general mesh and add in the resolution, with T-Junctions, after converting to T-Splines. I chose to utilize both methods in separate areas of the face (**figure 8**). For areas such as the ears, I felt it was easier to make sure I had the resolution there beforehand; thus, you can see the groups of edge loops, made to be deleted after conversion. If you plan to use extraordinary points in an area, create them in polygons before you convert to T-Splines. Investing a bit more time in the planning stages will pay off—yielding nicer looking T-Junctions later on.

Some of the finer details around the eyes and mouth were added with T-Junctions after I converted the model to T-Splines. Adding this detail was very simple. I selected the edges (yellow) where I wanted to add control points (**figure 9**), selected "Insert Control Point," and a new partial isoparm was created that I could then manipulate for more detail above the eye. Since I had the "preserve surface" option on, inserting the new edge

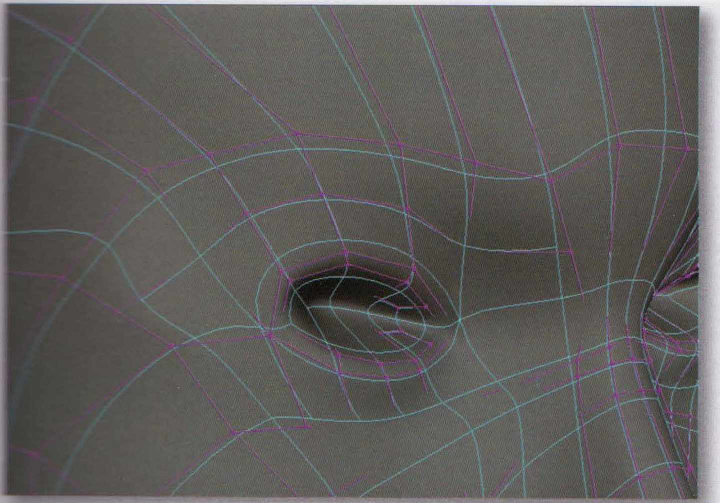


Figure 10–I now have a higher level of detail in this local area. T-Splines will not deform the area when adding resolution.

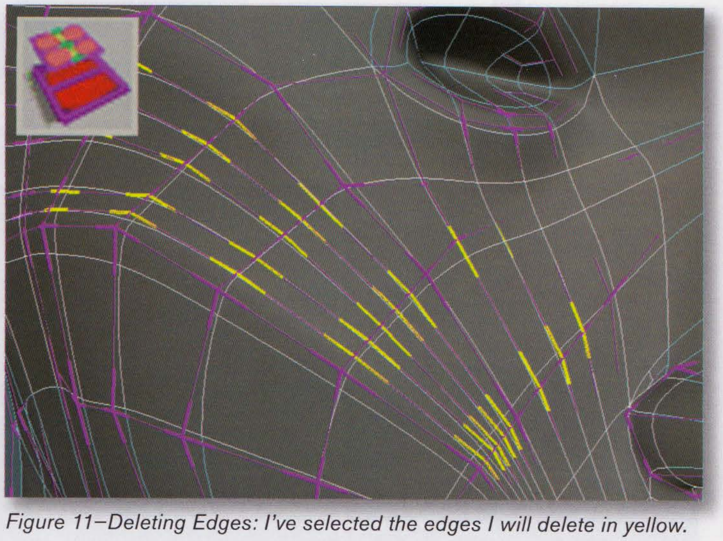


Figure 11–Deleting Edges: I've selected the edges I will delete in yellow.

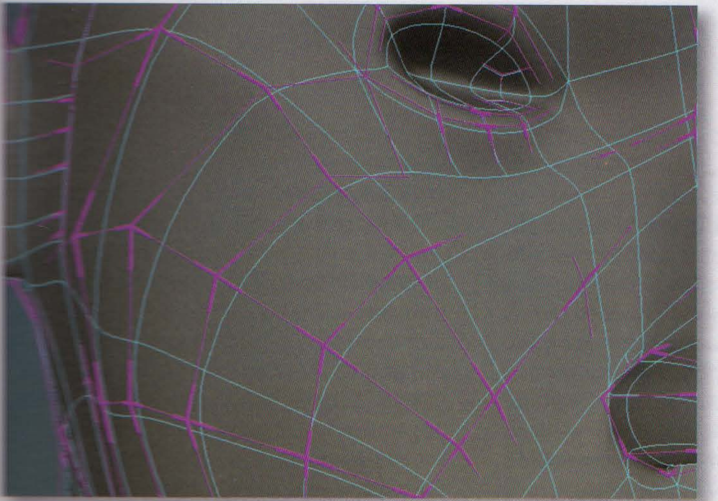


Figure 12–After selecting Delete Edges, my geometry is smoother and easier to work with.

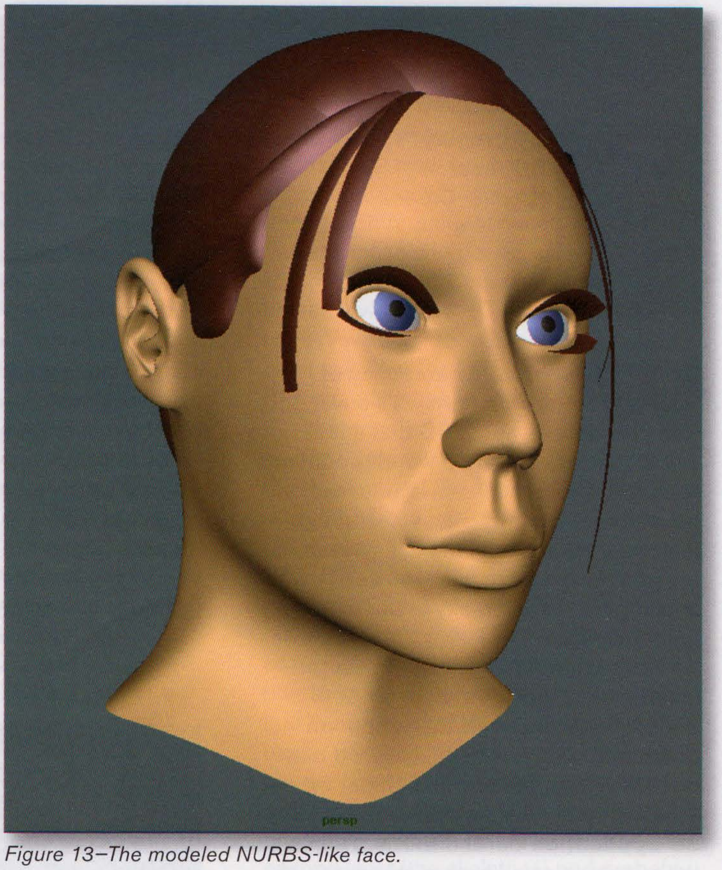


Figure 13–The modeled NURBS-like face.

didn't change the geometry at all and I was able to quickly and easily add the detail I wanted (**figure 10**).

I also chose to remove detail in the cheek to make a smoother, cleaner surface. This, of course, was done by deleting edges after conversion to T-Splines (**figure 11**). T-Splines will identify deleteable edges for you–by choosing "Show deleteable edges" all deleteable edges are indicated. You then simply select the edge(s) and use the "Delete edges" button (**figure 12**).

Modeling version 2 as a continuous surface for a NURBS pipeline was nice since I was able to have all of the benefits of a NURBS surface without having to worry about patches, seams, and lining up multiple control vertices (**figure 2**). Modeling the NURBS-flow version did take a bit longer than modeling the subd-flow version, and required more planning. However, in both versions, T-Splines allowed me the ability to add significant detail to parts of the model, while keeping the overall model nice and efficient (**figure 13**).

__Eric Allen__ graduated from Brigham Young University where he worked on the student Emmy winning Lemmings *short film. He currently works as the Modeling Lead for Daz Productions, a 3D content provider. Contact him at www.ericmatthewallen.com.*

real world production series

Part 1: NewTek LightWave 3D and Alias|MotionBuilder

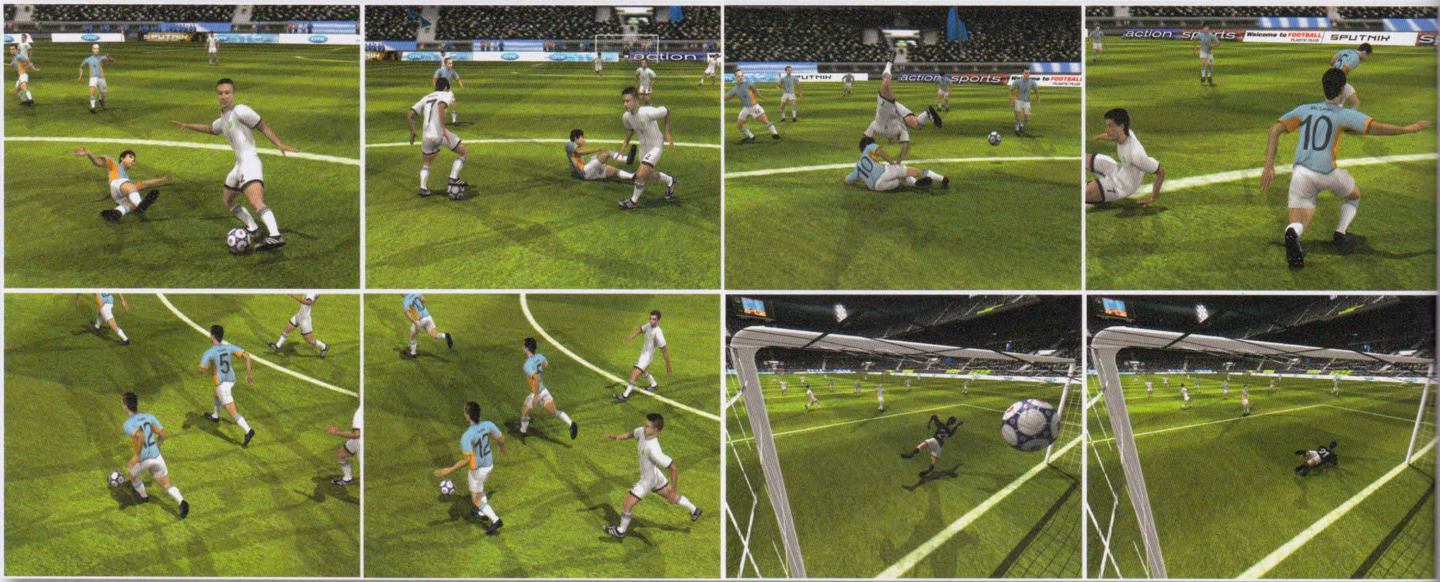


Figure 1–Screenshots from the project of this tutoria

By Gregory "T.Rex" Glezakos
3D Artist
Athens, Greece

Welcome to a new series of tutorials, where we'll cover issues about 3D that arise when dealing with a real world production situation. This series sometimes may be more of a step-by-step walkthrough than general tutorials, but in any case the goal would be to help you see how to accomplish certain tasks under extremely tight deadlines.

In this working environment we're using mainly **LightWave** for most of the 3D work; but with the exception of specific LightWave plugins we may deal with from time to time, most of these tutorials will be applicable to **any 3D application** out there. Also, since these tutorials will be based on actual, finished, and delivered projects we've made so far, they'll probably cover other applications as well beside 3D, like **Photoshop** or **Digital Fusion** and generally any program that exists in our arsenal and can help deliver a better job faster.

In our first tutorial of this series we'll see how we used motion capture files within **LightWave**, through the **FBX format** and **Alias MotionBuilder** software. So, let's get started...

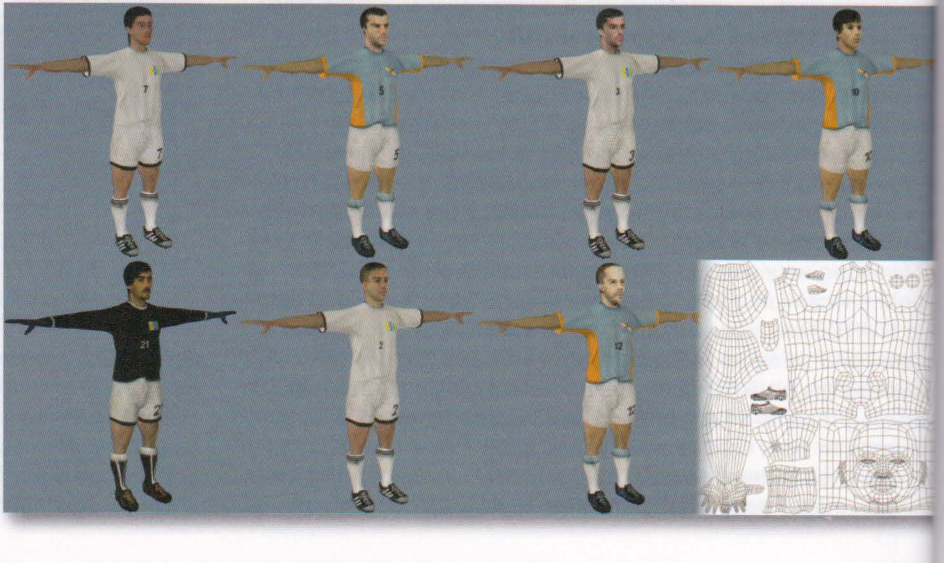
THE GOAL

This particular job (**figure 1**) required the creation from scratch of four shots, as if they were captured from within the *FIFA 2006 World Cup* Playstation video game, because paying license fees for using the actual video game for those shots was way more expensive than re-creating them in 3D. From concept to delivery these shots took **five days** in total, two people working and Motek (www.e-motek.com)– a great company in Holland that was responsible for the motion capture of the soccer players. Let's see how we deliver the shots in five days.

DAYS 1, 2, AND 3

The first day after we took on this job was a fun day, because we played a lot of *FIFA 2006*, trying to get the feel of how this game looks (yeah, right!) and also to make notice of what is seen from the angles the director wanted us to re-create. After we had the camera angles approved, we started the modeling of the stadium and the soccer players. One person was responsible for the modeling of the stadium and the other fo the players. Since the goal was to make the shots look like a **real-time playable game** and not like a rendered high

Figure 2–Player models and their texture



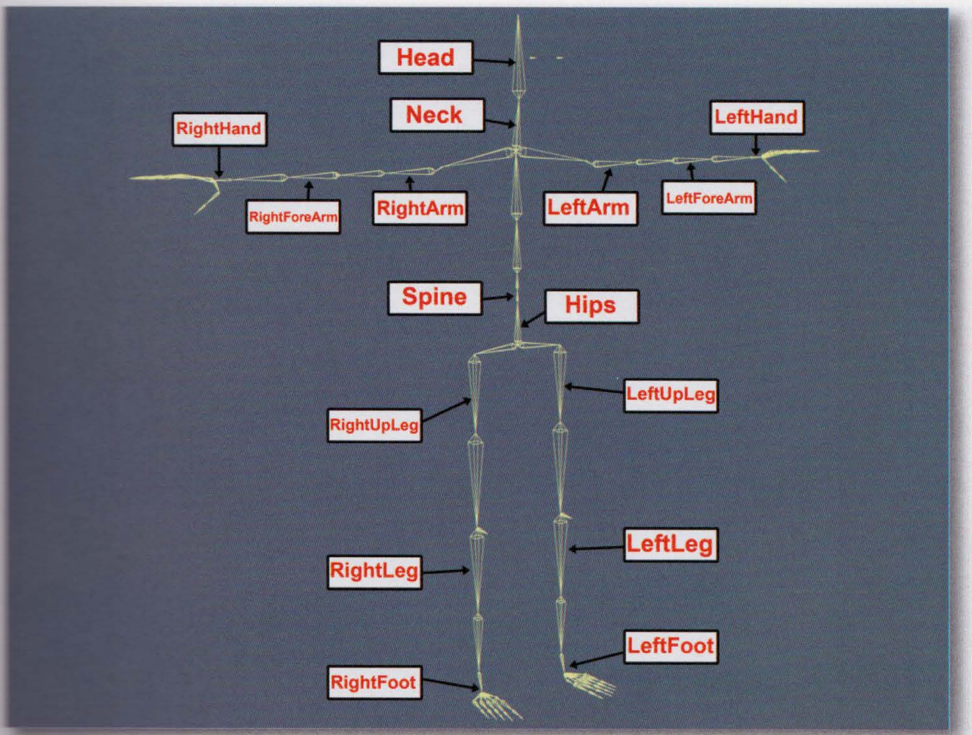


Figure 3–Skelegons with the most required bones, ready for MotionBuilder.

resolution cinematic video, we model everything with respect to that–low resolution models, based heavily on texture work. It took three days to finish the model of the stadium and two days for one character, but with **seven** different textures, **six** players, and **one** goalkeeper (**figure 2**). The rest of the players would be multiplied by those.

During the second day of work, **Motek** studio in Holland was contacted and asked for **motion capture sessions** regarding the needed motions the players had to perform. Two days later (the fourth day), the motion capture files that we needed were ftp'ed to us. The guys at Motek are real masters at their jobs and can deliver in no time, under any circumstance. Thankfully, we received more motion data than we'd asked for. This proved essential, as we had more options to choose from in order to better choreograph the soccer players.

DAY 4

Motek's motion capture pipeline is based around **Alias MotionBuilder** software, so the files we received were in **FBX format**. This brings us to the main purpose of this tutorial, which is how we can use these FBX motion capture files with LightWave. The following procedures refer to LightWave, but a **similar** process can be applied to **other 3D** applications as well. Of course, you'll need MotionBuilder Standard or Pro edition to use these FBX motion files, as we need to apply them to our characters first and then export them from MotionBuilder to our 3D app, again using the FBX format. Also, you'll need to download and install the latest version of the FBX plugin for LightWave, through the Alias web site: *http://www.alias.com/eng/products-services/fbx/download.shtml*.

You'll have to create a free account at the Alias site if you don't have one already, in order to access the download page. Depending on the LightWave version you have, you should download the appropriate FBX version; we used **LightWave 8.5** with the latest **2005.08 FBX** plugin version.

MODELER SETUP

So, the fourth day we received the mocap (motion capture) file, which was a single FBX file with 18 motion takes. Before we go to MotionBuilder, though, we have to prepare a scene in LightWave with a properly rigged character in the default T-Stance, which eventually will receive the mocap files. MotionBuilder can work with biped or quadruped rigs; for our case we'll use a biped rig setup, which we don't even have to create because David Spamer *(Contact@Pixeldimensions.com)* was kind enough to provide the LightWave community with a proper biped rig file. This is a **skelegon** object (LightWave's Modeler skeletons) that conforms with all the necessary requirements MotionBuilder has, in order for our characters to use the mocap files automatically without any extra effort. You can download this skeleton file from: *http://www.pixeldimensions.com/skeleton/3D_Dave_Skeleton_2.3.zip*.

You can use this rig as it is, or you can edit it as you like. You can re-adjust the bones to fit your character's T-Stance and you can add or delete bones. But whatever you do, you need to make sure that the bones seen below exist in your rig with this exact naming:

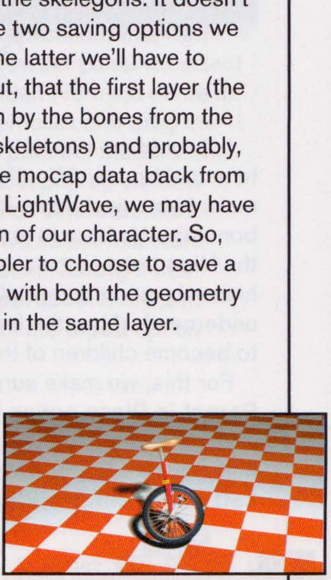
Hips, Spine, Neck, Head, RightArm, RightForeArm, RightHand, RightUpLeg, RightLeg, RightFoot, LeftArm, LeftForeArm, LeftHand, LeftUpLeg, LeftLeg, LeftFoot (**figure 3**).

This is one rule that MotionBuilder requires to make our life easier, because this way it will recognize our rig's structure automatically; otherwise, we'll have to manually map our bones to the correspondence bones MotionBuilder uses for characterization of the rig, which is an extra and tedious step we should avoid. For example, if we name the **RightArm** bone as **RightBicep,** MotionBuilder would not know what this bone does and we'd have to use the Mapping List to tell MotionBuilder where the **RightBicep** bone should correspond, which in this case would be the **RightArm** bone. But, if we name these bones as above, then MotionBuilder can automatically "characterize" them properly with a single click. The "**3D_Dave_Skeleton_Rig**" has all these bones named correctly, so we shouldn't have much to do with the skelegons, except slight adjustments to the bone tips to properly fit our character.

Also, notice another requirement of MotionBuilder: that our characters have to face towards the **+Z-axis**, which for LightWave corresponds to the **–Z-axis**; don't ask why. The "**3D_Dave_Skeleton_Rig**" already faces towards the –Z-axis, so if our character is facing the opposite way, we should rotate the character 180 degrees in the heading, not the skelegon rig, and then adjust the bone tips to fit the character.

LAYOUT SETUP

As soon as we've done that, we save our character with the skelegons, either as a single layer object or as a two layer object–one with the character's geometry and the other with the skelegons. It doesn't matter which of the two saving options we choose, but with the latter we'll have to make sure in Layout, that the first layer (the character) is driven by the bones from the second layer (the skeletons) and probably, when we merge the mocap data back from MotionBuilder into LightWave, we may have to fix the orientation of our character. So, it's easier and simpler to choose to save a single layer object, with both the geometry and the skelegons in the same layer.



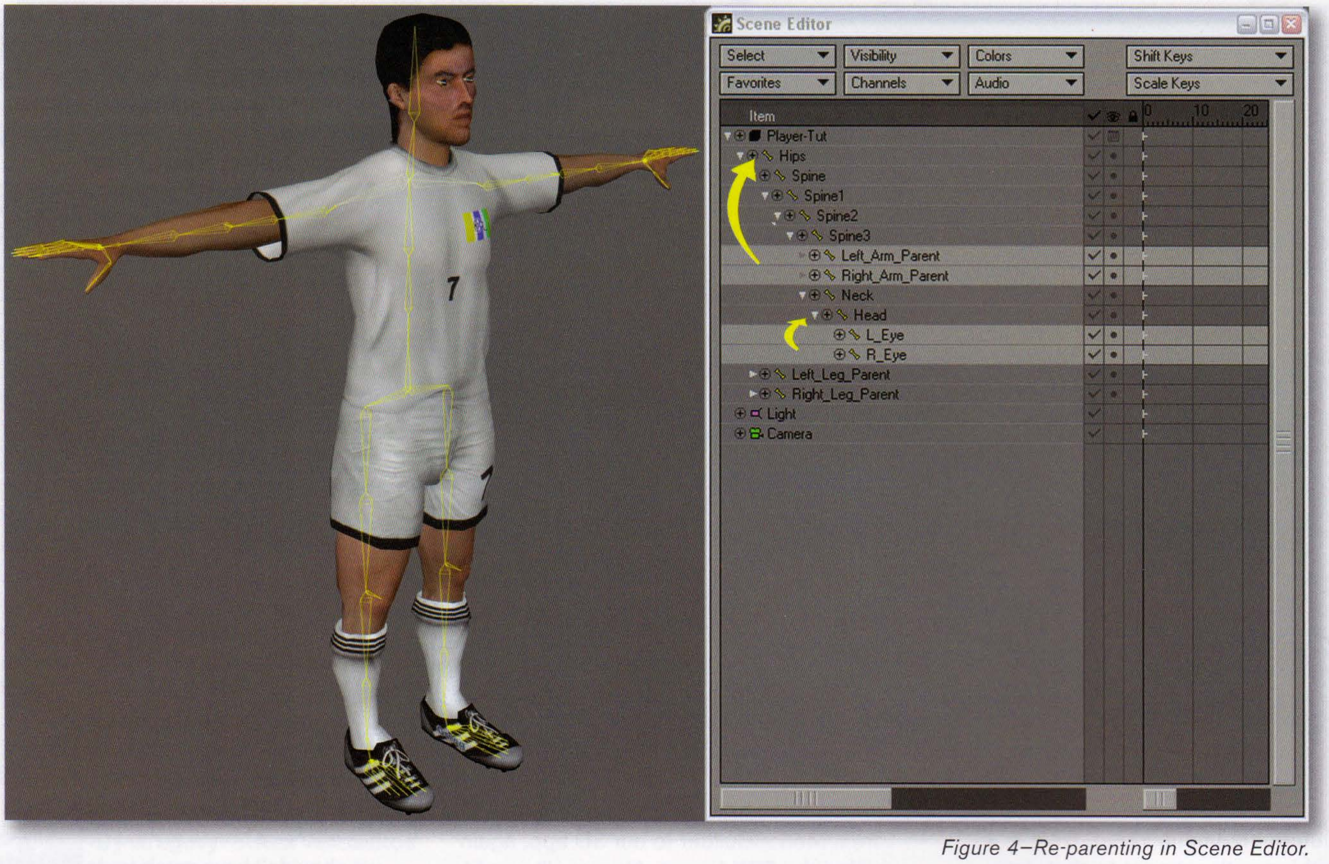

Scene Editor
Select
Visibility
Colors
Shift Keys
Favorites
Channels
Audio
Scale Keys
Item
0
10
20
Player-Tut
Hips
Spine
Spine1
Spine2
Spine3
Left_Arm_Parent
Right_Arm_Parent
Neck
Head
L_Eye
R_Eye
Left_Leg_Parent
Right_Leg_Parent
Light
Camera

Figure 4–Re-parenting in Scene Editor.

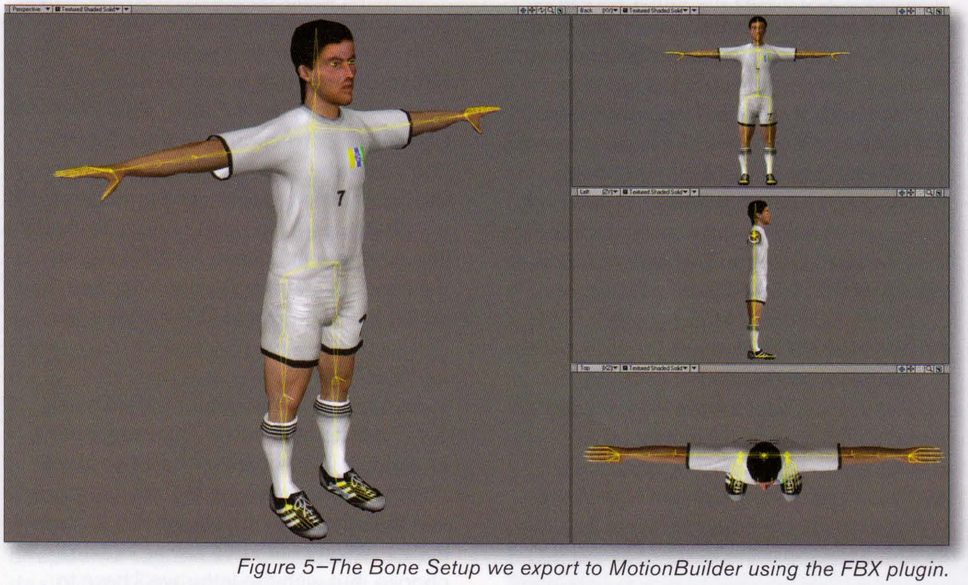
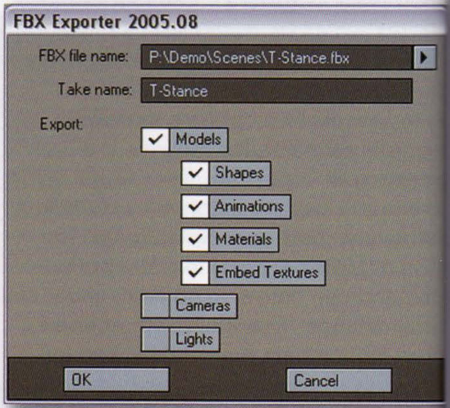


Figure 5–The Bone Setup we export to MotionBuilder using the FBX plugin.



FBX Exporter 2005.08

FBX file name: P:\Demo\Scenes\T-Stance.fbx

Take name: T-Stance

Export:

☑ Models

☑ Shapes

☑ Animations

☑ Materials

☑ Embed Textures

☐ Cameras

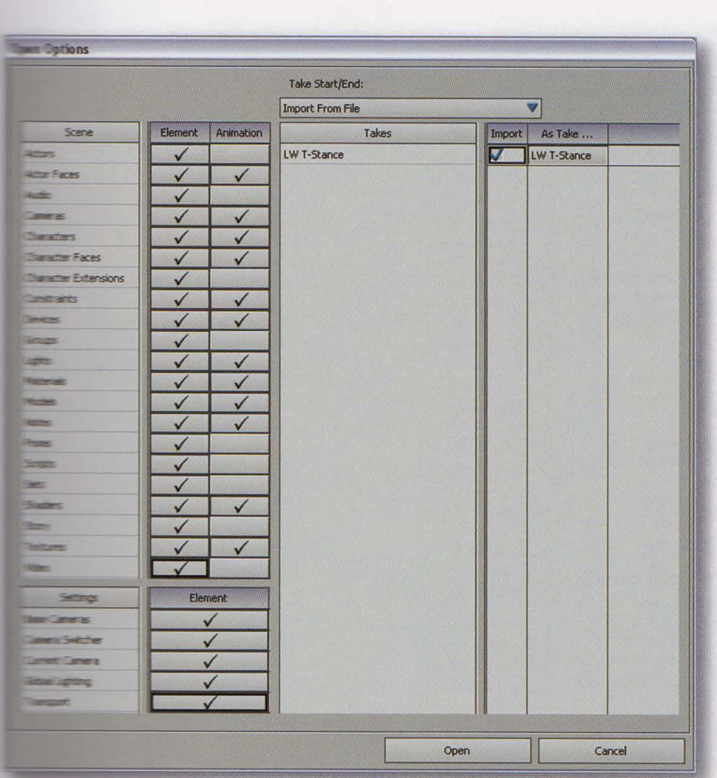
☐ Lights

Figure 6–The FBX Exporter options

In Layout we convert our skelegons to actual bones. After this, we'll have to make two adjustments in the generated bone rig. We have to make sure that the **Hips** bone is the root of our bone hierarchy and the rest of the bones fall underneath. Also, the two **Eye** bones have to become children of the **Head** bone.

For this, we make sure that the **Parent in Place** option is activated in the General Options panel. Then, in the Scene Editor, we drag the **LeftLegParent** and **RightLegParent** bones and drop them on the **Hips** bone, in order to make them both children of that bone. We do the same for the **L_Eye** and **R_Eye** bones, but this time we drop them on the **Head** bone, so they both become children of that bone (**figure 4**).

If you have painted **weight maps** for your character's bone structure and want to use them, make sure you apply those to the corresponding bones. We can do that either in Modeler or in Layout. For our case, we got away **without** painting any weights since the bone structure is very well laid out and we didn't want ultra realistic deformations. So we relied entirely on the bone influence ranges for the deformation of our characters. Either way, if there are no weight maps on the character, MotionBuilder's FBX format "converts" the default bone influence range to pretty decent actual weight maps and assigns them automatically on the character. Later, when merging the FBX motions in LightWave, it's up to the user to use them or remove them,



Open Options
Take Start/End:
Import From File
Open
Cancel

Scene	Element	Animation
Actors	✓	
Actor Faces	✓	✓
Audio	✓	
Cameras	✓	✓
Characters	✓	✓
Character Faces	✓	✓
Character Extensions	✓	
Constraints	✓	✓
Devices	✓	✓
Groups	✓	
Lights	✓	✓
Materials	✓	✓
Models	✓	✓
Notes	✓	✓
Poses	✓	
Scripts	✓	
Sets	✓	
Shaders	✓	✓
Story	✓	
Textures	✓	✓
Video	✓	

Settings	Element
Base Cameras	✓
Camera Switcher	✓
Current Camera	✓
Global Lighting	✓
Transport	✓

Takes
LW T-Stance

Import	As Take ...
☑	LW T-Stance

Figure 7–The dialog options for the opening of the LightWave scene in MotionBuilder.

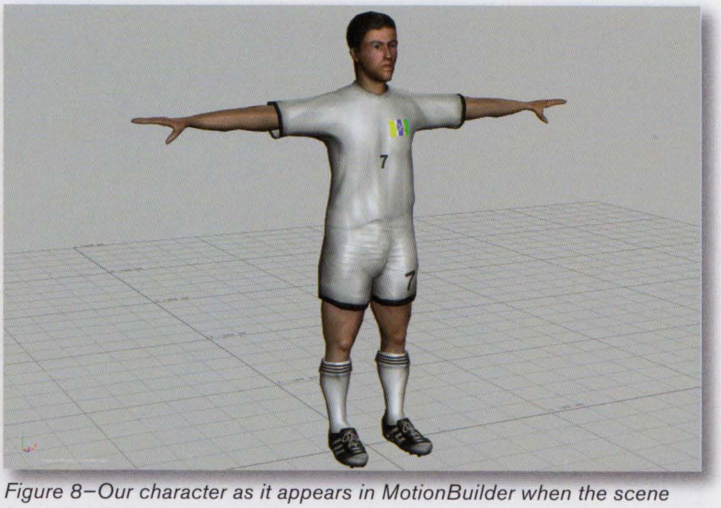
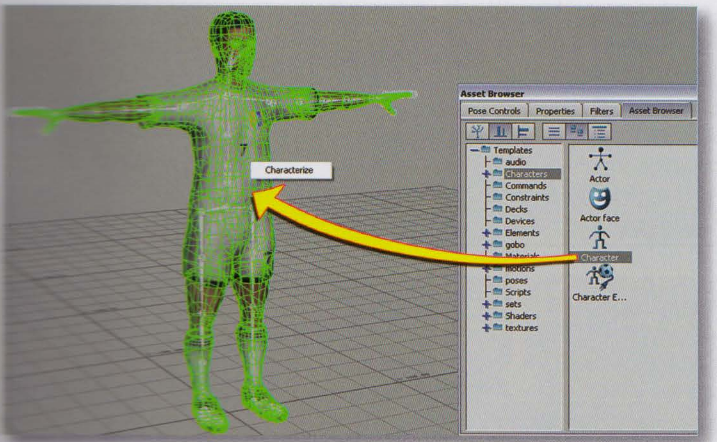
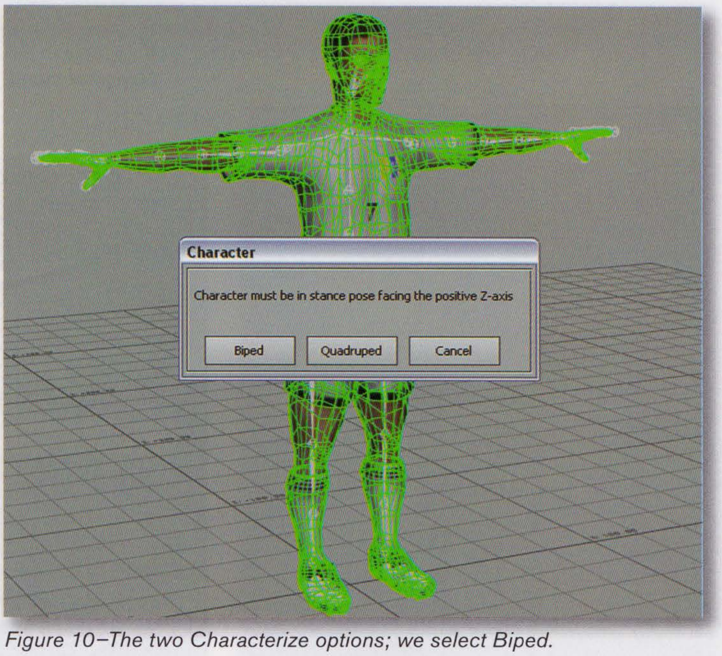


Figure 8–Our character as it appears in MotionBuilder when the scene loads the first time.



Characterize
Asset Browser
Pose Controls
Properties
Filters
Asset Browser
Templates
audio
Characters
Commands
Constraints
Decks
Devices
Elements
gobo
Materials
motions
poses
Scripts
sets
Shaders
textures
Actor
Actor face
Character
Character E...

Figure 9–Drag and drop the Character icon onto the model to Characterize.



Character
Biped
Quadruped
Cancel

Character must be in stance pose facing the positive Z-axis

Figure 10–The two Characterize options; we select Biped.

reverting to the default bone influence range without any weight maps applied.

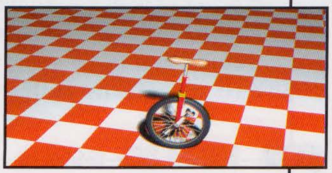
Now, we save this scene under the name "T-Stance," for example, as this will be our default scene when exporting our characters towards MotionBuilder and when merging back the animation to LightWave from MotionBuilder (**figure 5**).

The last thing to do in Layout is to export this scene using the FBX plugin, so we can transfer it to MotionBuilder. We choose the **FBX Export** option and we **uncheck** the **Cameras** and **Lights** since we don't care to export any of those. For a **Take** name we can leave the default, but it would make more sense in MotionBuilder to name this something like "T-Stance," since this is what this take is, after all. Last, we provide a valid file name for the Exporter, using the little black arrow at the top right; a "T-Stance" for the FBX scene name will also do fine (**figure 6**).

MOTIONBUILDER'S HEAVEN

In MotionBuilder we open the FBX scene we've just exported from Layout. A panel pops up; we should leave everything as it is and click the **OK** button (**figure 7**). A moment later our character is loaded along with its skeleton structure (**figure 8**).

The first thing to do is to characterize our object; in other words, make our character's rig MotionBuilder friendly. From the **Asset Browser** panel we select the **Characters** folder. From the contents of that folder, we left-click and drag the **Character** symbol onto our model in the OpenGL view port. A pop-up window shows with the word "**Characterize**" (**figure 9**). When we select that word another window opens, asking if this is a **biped** or a **quadruped** character; obviously, we select the **biped** option (**figure 10**).



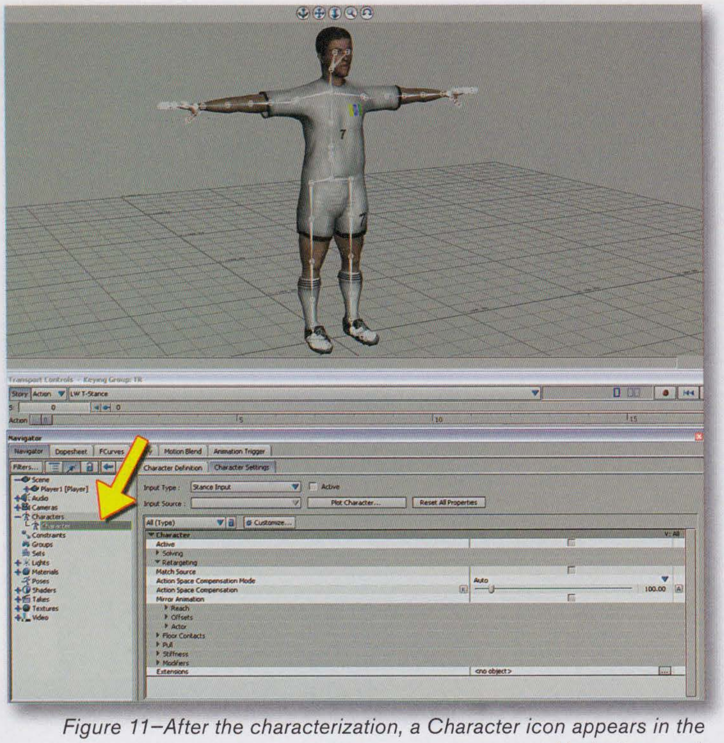
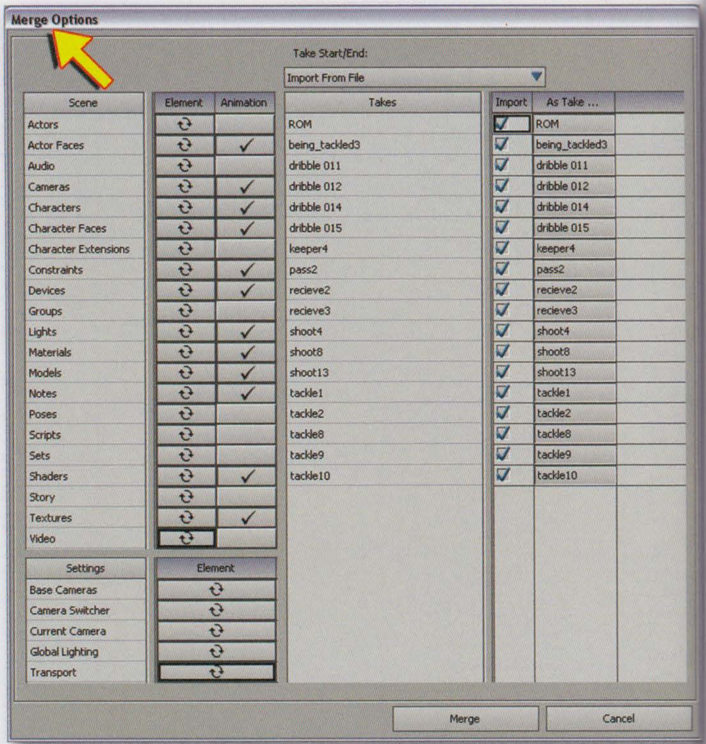


Figure 11–After the characterization, a Character icon appears in the Navigator panel.



Merge Options

Take Start/End: Import From File

Scene	Element	Animation
Actors	↻	
Actor Faces	↻	✓
Audio	↻	
Cameras	↻	✓
Characters	↻	✓
Character Faces	↻	✓
Character Extensions	↻	
Constraints	↻	✓
Devices	↻	✓
Groups	↻	
Lights	↻	✓
Materials	↻	✓
Models	↻	✓
Notes	↻	✓
Poses	↻	
Scripts	↻	
Sets	↻	
Shaders	↻	✓
Story	↻	
Textures	↻	✓
Video	↻	

Settings	Element
Base Cameras	↻
Camera Switcher	↻
Current Camera	↻
Global Lighting	↻
Transport	↻

Takes	Import	As Take ...
ROM	☑	ROM
being_tackled3	☑	being_tackled3
dribble 011	☑	dribble 011
dribble 012	☑	dribble 012
dribble 014	☑	dribble 014
dribble 015	☑	dribble 015
keeper4	☑	keeper4
pass2	☑	pass2
recieve2	☑	recieve2
recieve3	☑	recieve3
shoot4	☑	shoot4
shoot8	☑	shoot8
shoot13	☑	shoot13
tackle1	☑	tackle1
tackle2	☑	tackle2
tackle8	☑	tackle8
tackle9	☑	tackle9
tackle10	☑	tackle10

Figure 12–The Merge Options panel from the FBX file with all the mocap data.

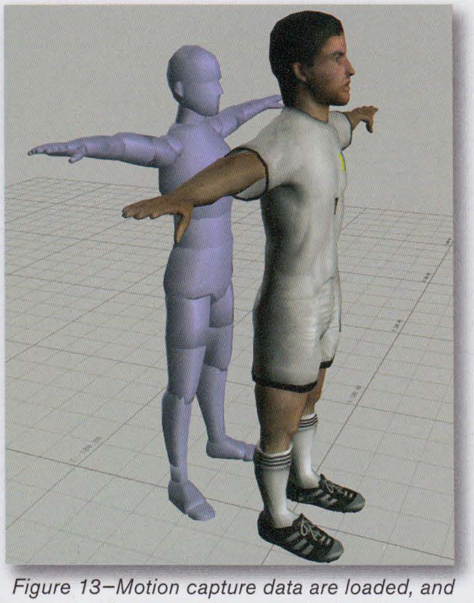
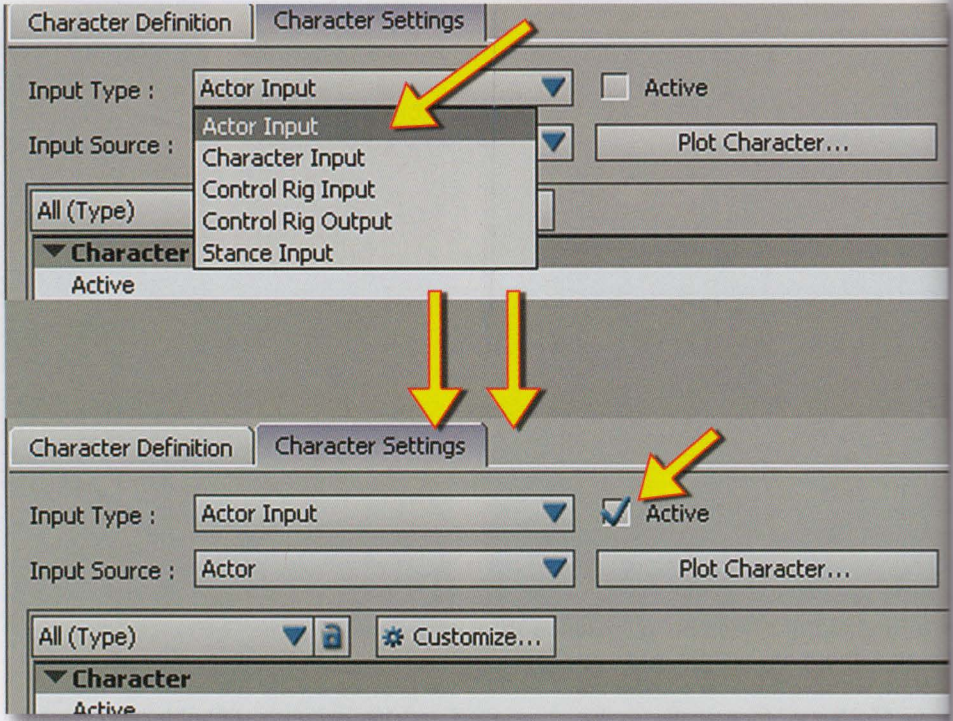


Figure 13–Motion capture data are loaded, and an Actor model appears in the view port.



Input Type : Actor Input

Input Source : Actor

☑ Active

Figure 14–Change the Character Input Type from Stance to Actor and activate it.

From this moment on, MotionBuilder **recognizes** our skeleton and new options are available for us to animate our character, one of them being the use of motion capture data. We can see in the **Navigator Panel** that a new field is created called "**Characters,**" under which we see a little humanoid icon also called **Character** (**figure 11**). This is where we set the various options we have to control and animate our model. If all went well so far, we can save our scene, under a new name, like "MB-Setup," just to be safe. What we do next is to **Merge** in this scene, the FBX scene we have with the motion capture data. We make sure we select from the **File** menu, the **Merge** option and **not** the **Open** or **Import** option (**figure 12**).

We click on the **OK** button to merge all takes–assuming our FBX file to have more than one take–and after a moment we see beside our character model, another humanoid model in the scene; this is the **Actor** model, which caries all the motion capture data, in multiple takes for our case (**figure 13**). Now, depending on the source of your FBX file, you may end up either with an **Actor,** or a **Control Rig**; we can control our character with either, but for our case we have an Actor with the mocap data, which should be the better option. The next thing to do is to go to the Navigator Panel and, under the **Characters** field, select the **Character** option; at the **Character Settings** at the

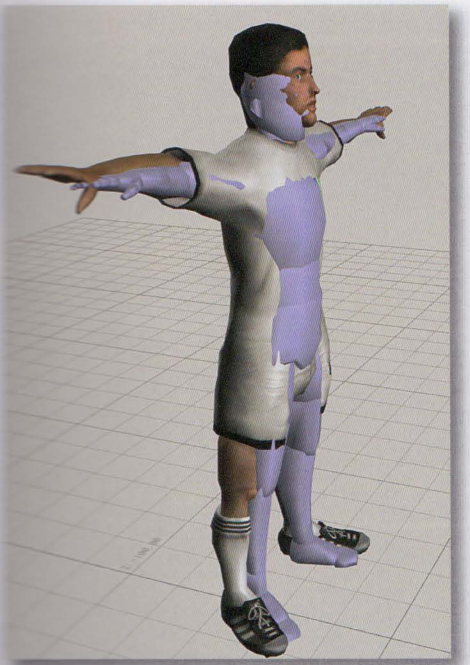
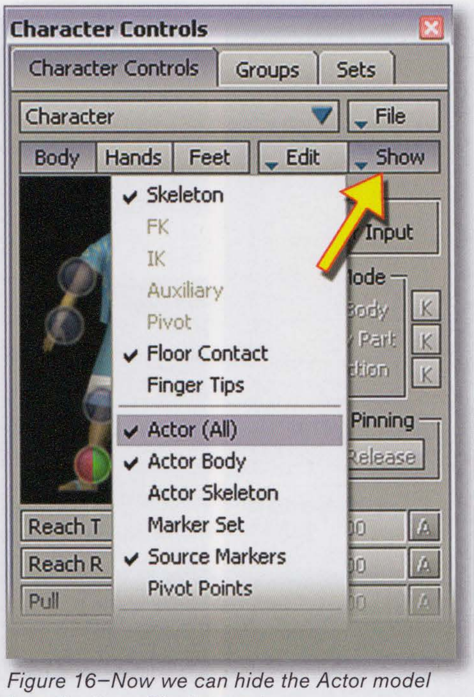


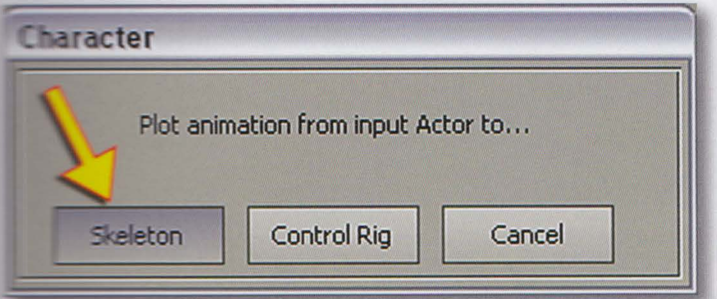
Figure 15–Our character "takes" proportionally the Actor's motions.



✓ Skeleton
FK
IK
Auxiliary
Pivot
✓ Floor Contact
Finger Tips
✓ Actor (All)
✓ Actor Body
Actor Skeleton
Marker Set
✓ Source Markers
Pivot Points

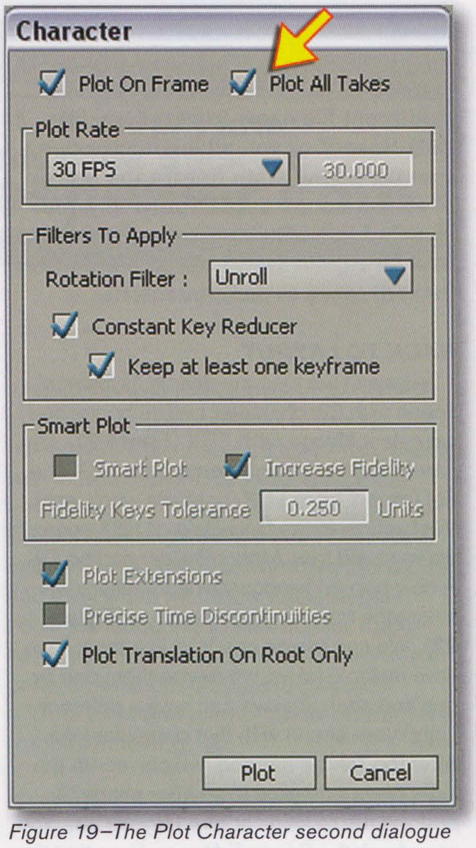
Figure 16–Now we can hide the Actor model from the OpenGL view port.

Figure 17–Our character in full motion heaven.



Plot animation from input Actor to...

Figure 18–The Plot Character first dialogue options.



☑ Plot On Frame ☑ Plot All Takes

Plot Rate: 30 FPS 30.000

Filters To Apply
Rotation Filter : Unroll
☑ Constant Key Reducer
☑ Keep at least one keyframe

Smart Plot
☐ Smart Plot ☑ Increase Fidelity
Fidelity Keys Tolerance 0.250 Units

☑ Plot Extensions
☐ Precise Time Discontinuities
☑ Plot Translation On Root Only

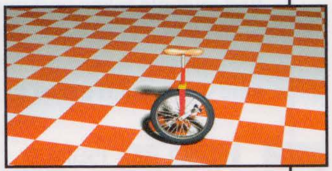
Figure 19–The Plot Character second dialogue options.

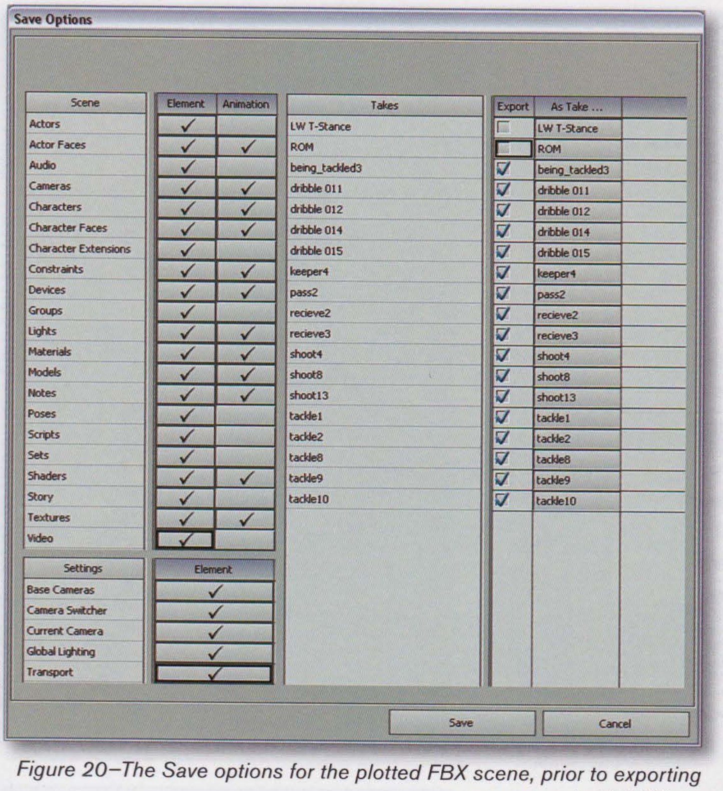
right panel, change the **Input Type** from Stance to **Actor** and check the **Activate** option next to this (**figure 14**).

Suddenly our character jumps over the Actor model, as it takes proportionally all of its motion data (**figure 15**). Now we can hide the Actor model and leave only our player character visible. We can do that at the **Character Controls** under the Show tab (**figure 16**). OK! We can select each take, hit the **Play** button, and watch our character perform the motions in real time (**figure 17**).

The last thing we have to do before export back to LightWave is bake all the mocap data to the skeleton bones. The "bake" term we're used to in LightWave, translates to "Plot" in MotionBuilder. So, we do this by clicking on the "**Plot Character...**" button located next to the **Input Source** option, at the **Character Settings** in the **Navigation Panel**. Then we choose the **Skeleton** to plot all the data (**figure 18**). Notice here one of the great features MotionBuilder has: we can keyframe animate **on top** of the mocap data, without losing or ruining anything, by using **Animation Layers**, similar to the layers Photoshop has. To do that, we Plot the animation from the input Actor to a **Control Rig** instead of the skeleton bones, then we can add as many animation layers on top as we like. But when we want to export animation from MotionBuilder to another 3D application through the FBX format, we have to Plot the animation from the input Actor to the skeleton. So, after we select the Skeleton to plot the data, another window pops up with the Plot Settings. Here we make sure to **Plot All Takes**, if we have more than one, and select the appropriate **Plot Rate**, which is the frame rate we want to use for our characters; the defaults for the rest of the settings are fine (**figure 19**).

Depending on how many takes we have, after a while the Plot is finished and we can go to ***File > Save*** to save





Save Options

Scene	Element	Animation
Actors	✓	
Actor Faces	✓	✓
Audio	✓	
Cameras	✓	✓
Characters	✓	✓
Character Faces	✓	✓
Character Extensions	✓	
Constraints	✓	✓
Devices	✓	✓
Groups	✓	
Lights	✓	✓
Materials	✓	✓
Models	✓	✓
Notes	✓	✓
Poses	✓	
Scripts	✓	
Sets	✓	
Shaders	✓	✓
Story	✓	
Textures	✓	✓
Video	✓	

Settings	Element
Base Cameras	✓
Camera Switcher	✓
Current Camera	✓
Global Lighting	✓
Transport	✓

Takes	Export	As Take ...
LW T-Stance	☐	LW T-Stance
ROM	☐	ROM
being_tackled3	☑	being_tackled3
dribble 011	☑	dribble 011
dribble 012	☑	dribble 012
dribble 014	☑	dribble 014
dribble 015	☑	dribble 015
keeper4	☑	keeper4
pass2	☑	pass2
recieve2	☑	recieve2
recieve3	☑	recieve3
shoot4	☑	shoot4
shoot8	☑	shoot8
shoot13	☑	shoot13
tackle1	☑	tackle1
tackle2	☑	tackle2
tackle8	☑	tackle8
tackle9	☑	tackle9
tackle10	☑	tackle10

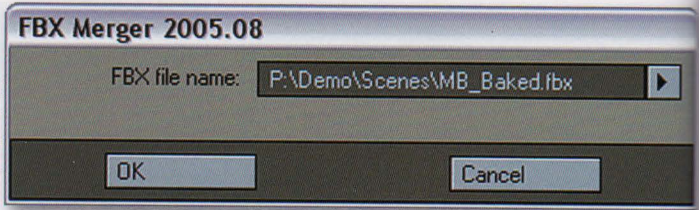
Figure 20–The Save options for the plotted FBX scene, prior to exporting to LightWave.

this scene in the **FBX** format; we use a different file name from before, like "MB-Baked" or something, so we don't lose the scene state **before** the **Merge** of all the mocap data (**figure 20**). Our work in MotionBuilder is done. We'll go back in LightWave to merge our plotted (baked) takes to our characters.

BACK TO LAYOUT

In Layout we load the default "T-Stance" scene with the character and its bones and we select **Merge with FBX** (**figure 21**). Be careful here to select **Merge with FBX** and *not* **Import from FBX,** or else it won't work. We browse to the "MB-Baked" FBX scene we've saved from MotionBuilder and hit OK. A new pop-up window appears where we select the take we want to merge (**figure 22**). We can merge one take at a time; if we have multiple takes, we merge them one by one and each time we can save a different LightWave scene with that particular take. After the Take is merged, we can scrub the LightWave's timeline to see our character perform the motion capture (**figure 23**). You can check the Scene Editor and see that the bones of the character have all the mocap keyframes applied properly. That is all there is about the process to bring mocap data from MotionBuilder into LightWave through the use of the FBX plugin and MotionBuilder.

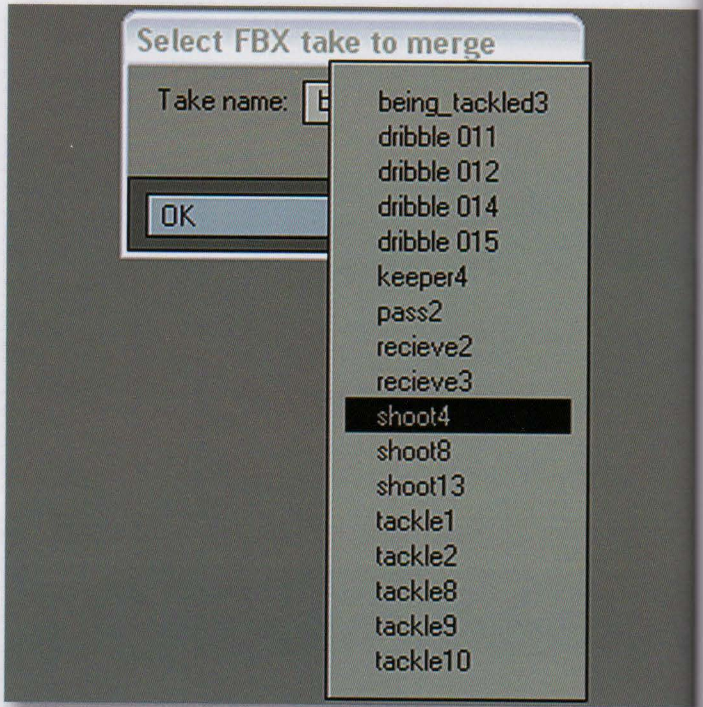
For our particular project, because we had multiple takes and multiple characters at the same time for our shots, to simplify the process we did the following. Each time we merged a take to the "T-Stance"



FBX Merger 2005.08

FBX file name: P:\Demo\Scenes\MB_Baked.fbx

Figure 21–The FBX Merger options in LightWave



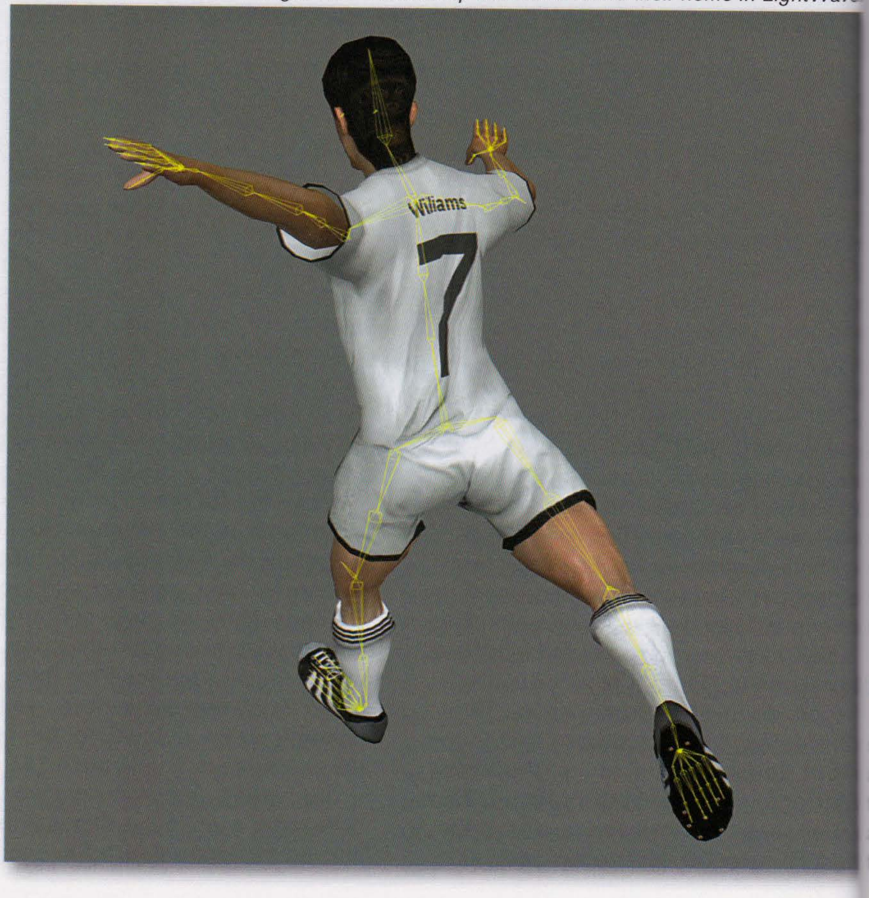
Select FBX take to merge

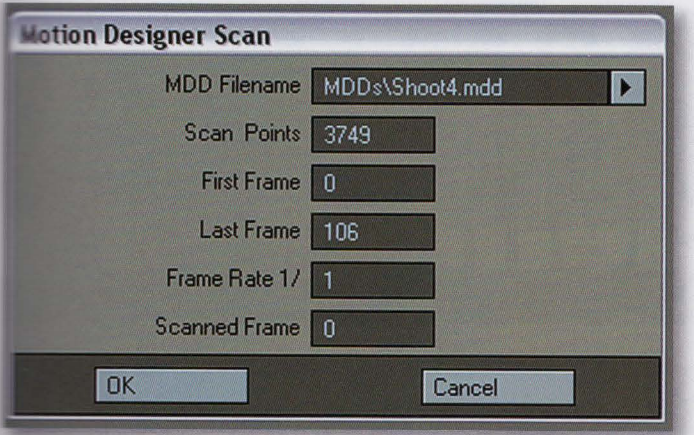
Take name:

- being_tackled3
- dribble 011
- dribble 012
- dribble 014
- dribble 015
- keeper4
- pass2
- recieve2
- recieve3
- shoot4
- shoot8
- shoot13
- tackle1
- tackle2
- tackle8
- tackle9
- tackle10

Figure 22–We can select one take at a time to merge

Figure 23–The mocap data have found their home in LightWave





Motion Designer Scan

MDD Filename: MDDs\Shoot4.mdd

Scan Points: 3749

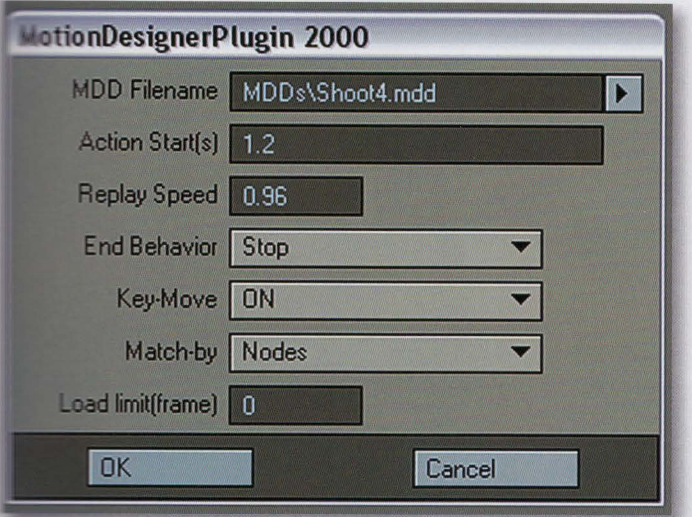
First Frame: 0

Last Frame: 106

Frame Rate 1/: 1

Scanned Frame: 0

Figure 24–The MDD Scan plugin options for baking a mocap deformation.



MotionDesignerPlugin 2000

MDD Filename: MDDs\Shoot4.mdd

Action Start(s): 1.2

Replay Speed: 0.96

End Behavior: Stop

Key-Move: ON

Match-by: Nodes

Load limit(frame): 0

Figure 25–The MDD Plug plugin options for loading a mocap deformation.

Without ResPower Super/Farm™

With ResPower Super/Farm™

3.2 THz Render Farm
Take Your Life Back At:
www.respower.com

scene in LightWave, we then used the **MDD Deformation Baking plugin** to bake all the mocap data into discreet **MDD** files (**figure 24**). This way we could use the players **without** the skeletons and all their mocap keyframed data. And also, since for all of our players the base model was the same object with only different textures, we ended up with as many baked **MDD** files as the mocap takes we wanted to use, which we could use freely for any of the seven different textures but with the same number of vertices player models. Using MDD files is much, much faster and lighter than having all those characters carrying their skeletons with all the mocap data, plus we can control when the motions start and the speed at which they will run (**figure 25**). And with that, the fourth day ended.

DAY 5

The last day prior to delivering the four shots, we choreographed all the players and the final staging of the shots we had to deliver. Since we had everything "baked and ready," setting the players to do their thing took no more than a couple of hours. The football object was keyframed by hand, after the player's movement was finished. Then the stadium was brought inside these scenes, and finally we started the rendering process. Rendering was like real time, since we didn't use high levels of anti-aliasing and had no motion blur; this had to look like a game capture, remember?

By the end of the fifth day, we delivered the four shots and the clients were very happy with the results; they didn't made a single change. You can download and watch the four shots at this link: http://www.sub-pixel.com/OTE-FIFA_Final.mov

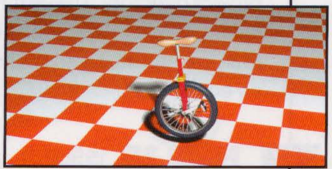
CONCLUSION

Alias MotionBuilder is probably one of the best applications available when it comes to biped or quadruped character animation, because it's very well laid out, extremely stable, and its **OpenGL performance** is umber fast and quick, which is the **number one demand** for character animation; for the same take, with similar sized OpenGL view ports and only the player model visible, MotionBuilder plays it back in 152-160 FPS, while LightWave plays it back in 32-36 FPS. In the same workstation! Not to mention that nowadays it's the **standard tool** when dealing with motion capture hardware equipment.

As you see, the process for sending scenes back and forth between LightWave and MotionBuilder is pretty straightforward and easy when you follow these steps. I hope you found this tutorial helpful. If you have any problems or comments, feel free to contact me at Gregory@sub-pixel.com.

Until next time, keep those characters (mocap) moving!

Gregory Glezakos *is an experienced 3D artist working professionally more than 12 years, either as a freelancer or contracted to major domestic production facilities. Today he is the Founder of the Sub-Pixel Studios (www.sub-pixel.com), a small but flexible post-production house. When between jobs, he teaches LightWave 3D and Digital Fusion to students.*



FX Magic with Maya and particleIllusion

By Eric Keller
Maya Animator
West Hollywood, California

Recently I've done several professional projects involving particle simulations and animations created in Maya, as well as adding particle effects to footage from film. Now I'm certainly no stranger to Maya's powerful dynamics module and all the fun stuff that can be created with particle expressions, fields, and so on. However, there are some situations in which Maya's particle workflow can be overkill. In these cases I've started to rely on a little program by wondertouch called particleIllusion.

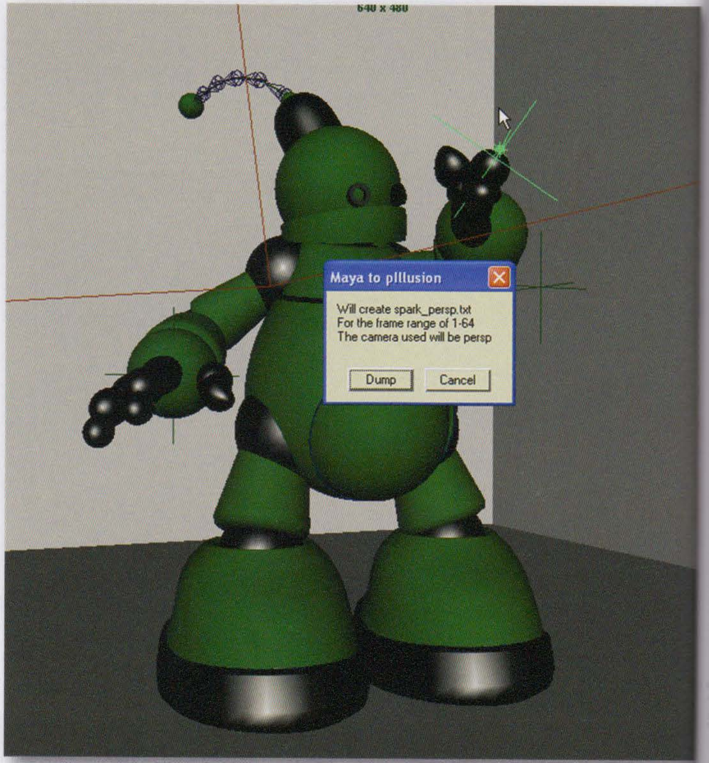
particleIllusion is a stand-alone 2D animation package devoted to creating particle effects. It's one of those programs that is so much fun to play with it reminds you why you got into computer graphics in the first place. It's extremely easy to use yet quite powerful. Although it is a 2D animation package, it is not very hard to simulate 3D effects. With the use of some free scripts available on the wondertouch site *(www.wondertouch.com)*, you can quickly incorporate the effects you create in particleIllusion into animations rendered with the more popular 3D packages such as Maya, LightWave, XSI, Max, and Cinema 4D.

I've used particleIllusion to create fireworks, comet trails, chilly wisps of breath, and snow. It's a very fast program that allows you to view results in real time. If you've ever had a situation in which you wanted to change the look of an animation while a client stands over your shoulder (I realize no one really wants to be in this situation, but it does happen!), this program will allow you to do just that–and make you look like a miracle worker in the process.

This tutorial will demonstrate how to use a MEL script to composite effects created in particleIllusion with an animation rendered in Maya. You can download a demo of the program from the wondertouch website at *www.wondertouch.com*. You'll also find the Position Exporter scripts for Maya, Lightwave, XSI, and Cinema4D, as well as the particleIllusion Exporter script for 3ds max, at the bottom of their download section (*www.wondertouch.com/downloads.asp*).

If you've never used particleIllusion before, I recommend playing with it a bit and looking through the HTML help guide that comes with the program. The tutorials are very well written and can get you up to speed after just a couple hours of fooling around. But if you want to dive right in with this tutorial, I've included the scene files and some QuickTime movies on the *HDRI3D* website (*www.hdri3d.com/resources*).

The MEL script I'm using is called maya2pi.mel, and it is available for download from the wondertouch downloads website (*www.wondertouch.com/downloads.asp*). It's very simple to use, but you will need to copy it into your scripts directory and source it before execution. The maya2pi.mel script will create a text file in the current project directory. This text file consists of the positional data for a selected object in a scene as it relates to the current camera. This can then be imported into particleIllusion and converted into keyframes for an emitter.



```
Will create spark_persp.txt
For the frame range of 1-64
The camera used will be persp
```

Figure 1—With the locator selected in the perspective camera, the maya2pi.mel script is activated. A text file called spark_persp.txt will be written to the project directory

I created a scene with a simple 64-frame animation of a robot whose finger has caught fire. He's shaking his hand in an attempt to put it out. As scenes go, it's no *Hamlet* but it should demonstrate the workflow well enough. A QuickTime of the animation before and after it's had particles added to it is available at the resources section of the *HDRI3D* website (*http://hdri3d.com/resources*). Here's how I used the script:

1. I created a locator called "spark" and parented it to the index finger on the robot's hand.

2. I also created a point light with a high intensity and quadratic fall off. This is also parented to the same finger.

3. I added some keyframes to the perspective camera. The maya2pi script will write out the positional data of the locator in relation to a selected camera so that the emitter I add in particleIllusion will match up perfectly.

4. I selected the spark locator and middle-mouse-clicked in the perspective window to make sure that it was the active camera. With the maya2pi.mel script in my scripts directory, I typed "maya2pi;" in the command line. A warning box popped up letting me know that a text file called spark_persp.txt would be written in my project directory (**figure 1**).

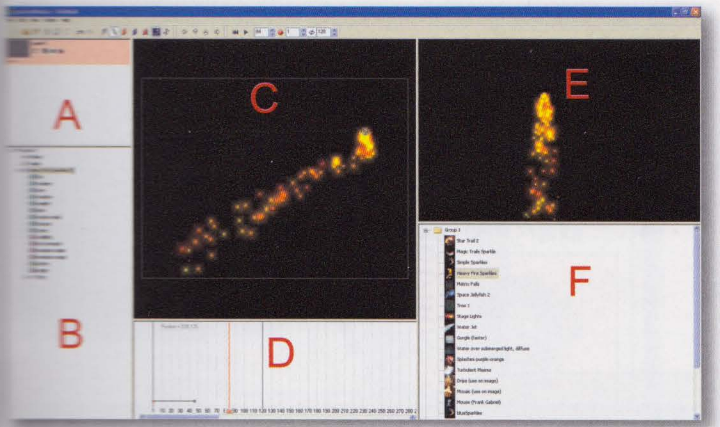


Figure 2–A quick tour of the particleIllusion workspace.

5. I rendered out the animation from Maya as a targa sequence at 640x480. I will bring this into particleIllusion as a background.

6. I saved my Maya scene and opened up particleIllusion.

Here's a quick tour of the particleIllusion interface (**figure 2**): The interface consists of 6 frames and a standard menu bar. From left to right the frames are divided into:

A - A layer palette, similar to display layers in Maya.
B - A hierarchy palette, similar to Maya's Outliner.
C - The stage area where the actual animating takes place.
D - The timeline/graph editor.
E - The preview window.
F - The library.

7. From the view window on the tool bar, I selected the Project Settings. This opened up a dialogue box.

8. In the dialogue box I can set the frame rate, the stage size, field rendering, motion blur, background color, etc.

9. I just changed the stage size to 640 x480 and left everything else at the default. Pressing "OK" closed the window and changed the project settings.

10. In the layer window I right-clicked over layer0 and chose "rename" from the pop-up menu. I renamed the layer "background."

11. I right-clicked over layer0 in the layer window again and chose "background image." From the file browser window, I searched through my computer and found the frame sequence I rendered from Maya.

12. A window popped-up and asked if I wanted to load a sequence. I accepted this and set the frame loop range to 70 frames. I can preview the animation by hitting the play button.

13. From the group1 folder in the library window, I chose "Heavy Fire Sparkles." With the Show Particle tool selected and the timeline set to frame 1, I clicked on the stage to add the emitter.

14. In the hierarchy window I right-clicked over "Heavy Fire Sparkles" and chose *Import > position data*. From the pop-up window I picked tab-delimited-text(from file)."

15. In the file browser window I searched for the spark_persp.txt file in my project directory and selected it.

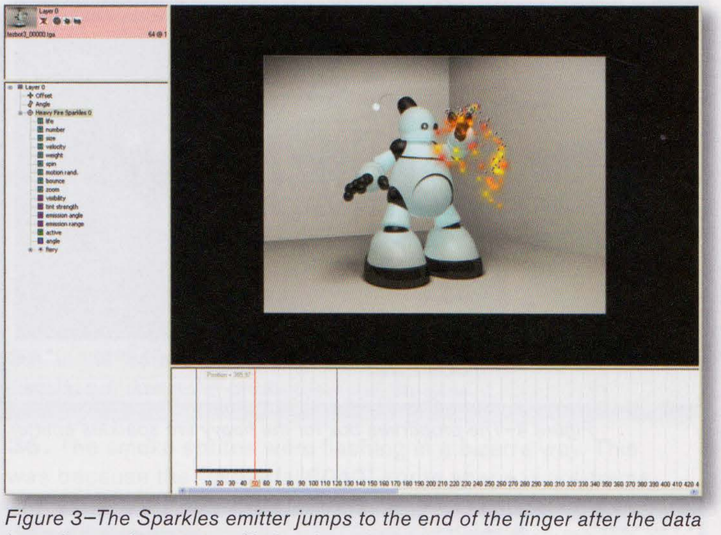


Figure 3–The Sparkles emitter jumps to the end of the finger after the data from the spark_persp.txt file has been imported.

16. When this file was imported, the emitter jumped to the end of the robot's finger. In the timeline, there is a keyframe set for every frame (**figure 3**).

17. When the animation was played, the emitter followed the robot's finger–looked pretty good! That was easy!

18. Then I wanted to change the look of the particles to a glowing blue shower of electric sparks. I right-clicked on the "Heavy Fire Sparkles" on the hierarchy window and opened up the properties window.

The properties window is like a secondary interface. From here you can change the image mapped to the particles. All particles in particleIllusion are sprites; you can use the images that come with the libraries or even create your own in Photoshop and import them. I've rendered out short sequences from Maya and brought them into particleIllusion as well.

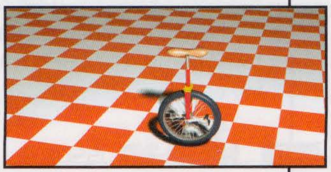
The properties window allows you to access all of the attributes as well as the timeline for your particle animation. Changes here will be updated on the stage when you press the "OK" button. The box has two main sections: properties and particles. Properties allows you to control some overall settings for the particle type. You can also add, duplicate, and delete particle types in the hierarchy window in this box.

The particles tab of the properties box consists of three sub-tabs: Behavior, Change Shape, and Colors. This section allows you to control the look of the individual particles. The behavior window has a timeline that changes depending on which attribute you are adjusting. The main timeline on the stage will update when you accept changes made in the properties box.

The properties dialogue box also has a preview window that will allow you to see what the changes you're making will look like without having to leave the properties box.

19. I switched to the color tab and changed the color gradient so that it started as a light blue, moved to a medium blue, and then ended at a dark blue.

20. I set the transparency gradient so that it moved from white to black. I set the gradient to "use full gradient (normal)."



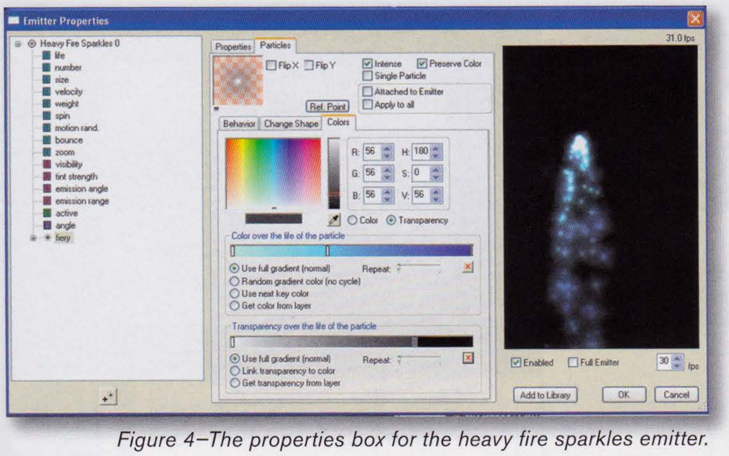

Emitter Properties
Heavy Fire Sparkles 0
life
number
size
velocity
weight
spin
motion rand.
bounce
zoom
visibility
tint strength
emission angle
emission range
active
angle
fiery
Properties
Particles
Behavior
Change Shape
Colors
Color over the life of the particle
Transparency over the life of the particle
Enabled
Full Emitter
Add to Library
OK
Cancel

Figure 4–The properties box for the heavy fire sparkles emitter.

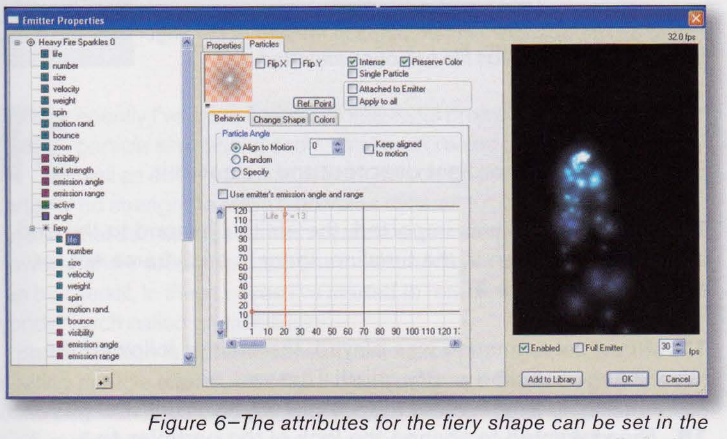

Emitter Properties
Heavy Fire Sparkles 0
fiery
Particle Angle
Align to Motion
Random
Specify
Keep aligned to motion
Use emitter's emission angle and range
Life
Enabled
Full Emitter
Add to Library
OK
Cancel

Figure 6–The attributes for the fiery shape can be set in the properties dialog box.

21. I turned the repeat slider on both the color and transparency all the way down to the left (**figure 4**).

22. In the Change Shape tab, I selected "blurred spikey star" and hit the "make active" button (**figure 5**).

23. I switched back to the properties tab in the properties box and set the following attributes for the fiery object. You may have noticed that in the hierarchy window there are attributes under "Heavy Fire Sparkles 0" and then a group called "fiery." When the fiery group is expanded, there's a whole new set of attributes, many of which are duplicates of those above. The reason for this is that it is possible to add more than one particle type to an emitter. So the first set of attributes under Heavy Fire Sparkles controls the attributes for all of the particles assigned to that emitter. Then, each particle type has a secondary set of controls so that the different particles belonging to that emitter can have different behaviors. This will make more sense when we add some smoke in a moment. For now I am only changing the settings under the "fiery" group. I'm leaving the attributes under "Heavy Fire Sparkles 0" alone so that they don't conflict with the particle type settings.

To input the settings, I select the attribute in the hierarchy window and in the timeline/graph editor (from the main layout or properties window) I move the keyframe at frame 1 up or down to the desired value. I can also set keyframes at a later frame if I want, and it will update the animation when I'm ready to apply my settings.

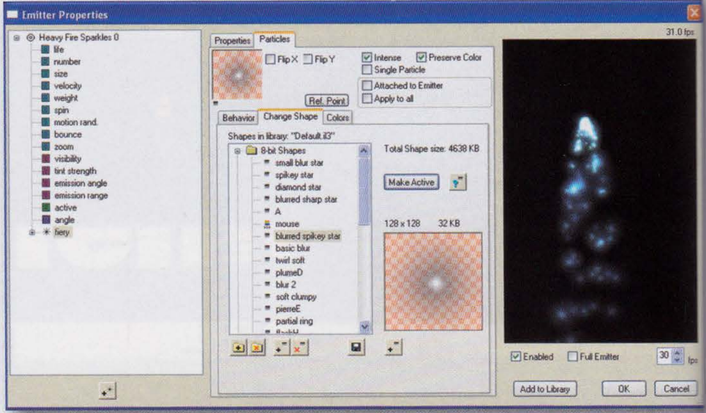

Emitter Properties
Heavy Fire Sparkles 0
Properties
Particles
Behavior
Change Shape
Colors
Shapes in library: "Default.il3"
8-bit Shapes
small blur star
spikey star
diamond star
blurred sharp star
A
mouse
blurred spikey star
basic blur
twirl soft
plumeD
blur 2
soft clumpy
pierreE
partial ring
Total Shape size: 4639 KB
Make Active
128 x 128
32 KB
Enabled
Full Emitter
Add to Library
OK
Cancel

Figure 5–A new shape is selected from the shape library for the particles.

Here are the values I selected for the fiery group(**figure 6**):

life=13
number=75
size=18
velocity=12
weight=8
spin=4
motion rand=5
visibility=100

For the variation attributes I created these settings:

life variation=39
number variation=5
size variation=5
velocity variation = 5
weight variation=3
spin variation=0
motion rand variation=8

24. The size over life attribute already has some keyframes set, so I edited these a little just to tweak the behavior of the particles.

25. I edited the velocity over life so that it moves from 100 to 40%.

26. I did the same with the weight over life.

27. I left the rest at their default settings, but I encourage you to play with them just to see what happens.

28. I clicked "OK" to apply the settings and close the properties box.

When the animation is played, the robot now emits blue sparks from the tip of his finger. I wanted to add some smoke to the emitter as well, so I opened up the properties box for the Heavy Fire Sparkles again.

I'm going to add a second particle to this emitter for the smoke. This part will show how I created a new particle type almost from scratch. The emitter will now shoot out two different particle types; the sparks and the smoke.

29. I opened up the properties dialogue box again. In the hierarchy section of the properties dialogue box, I selected "Heavy Fire Sparkles 0."

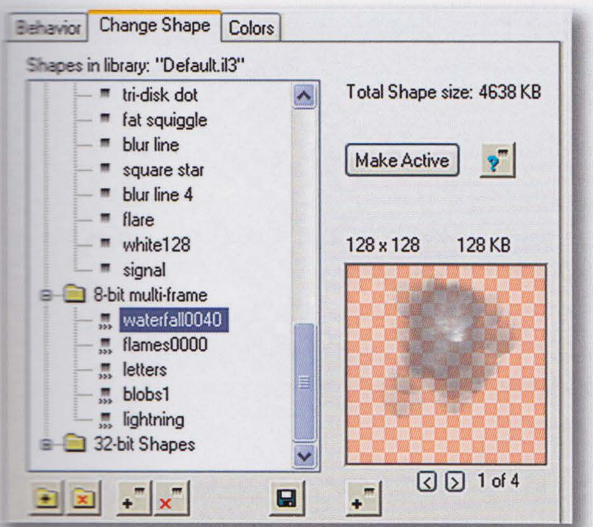


Figure 7—A multiframe shape is chosen from the shape library.

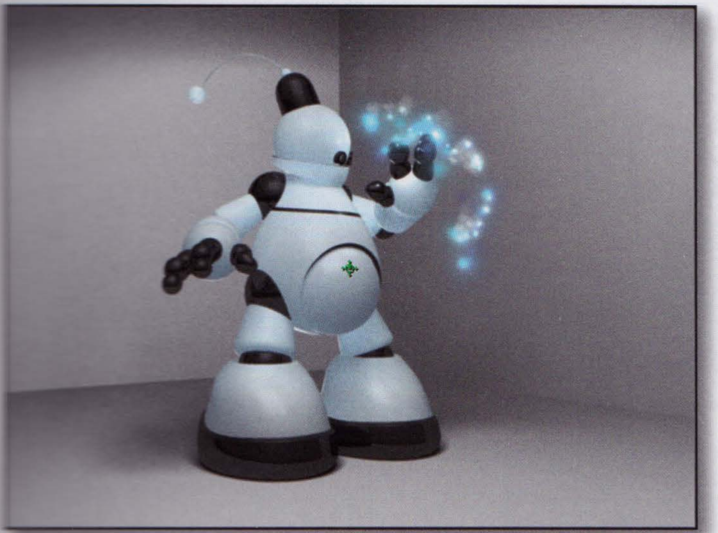


Figure 9—After the settings are applied, the robot has both smoke and sparks coming from his finger. Don't worry, he is not programmed to feel pain.

30. I clicked on the "Add Particle Type" button below the hierarchy window (it looks like a plus sign with a spark next to it).

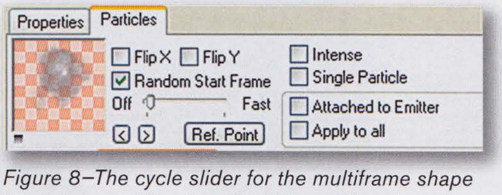
31. "New Particle Type" appeared above "Fiery" in the hierarchy window. I right clicked over it and chose "rename."

32. I changed the name to "smokey."

33. I right-clicked over it again and chose "Move down" so that it was behind the spark object.

When I dragged around in the preview window, I saw some very basic dots. This was pretty much a raw emitter, so I had to build the smoke up from scratch. After you do this a few times, it becomes pretty straightforward.

34. The first thing I wanted to do was change the shape into something smokier. In the properties tab I switched over to "Change Shape." From the list in the default library, I chose "Waterfall0040" from the 8 bit multiframe folder. Once I selected it, I hit the "Make Active" button and the preview changed (**figure 7**).



☐ Flip X ☐ Flip Y
☑ Random Start Frame
☐ Intense
☐ Single Particle
☐ Attached to Emitter
☐ Apply to all

Figure 8—The cycle slider for the multiframe shape is set to zero. "Random Start Frame" is selected.

35. In the hierarchy I expanded the attribute list under Smokey and started adjusting the attributes. I set the size at about 20.

36. The smoke sprites were flashing in a bizarre way. This was because the "Waterfall0040" sprite shape is a 4-frame animation and the sprites were cycling through it very quickly. What I wanted was for the sprites to not cycle through the animation, but for each sprite to pick a frame randomly from the cycle. To do this I just had to move the slider next to the preview window all the way to the left and select the "Random Start Frame" attribute (**figure 8**).

37. In the behavior tab I switched the particle's angle to "Align to Motion" and I deselected the option for "use emitter's angle and range."

38. In the hierarchy window I expanded the list of attributes below the smokey object and set the following parameters on the graph editor:

life=18
number=24
size=19
velocity=10
weight=-6
spin=17
motion rand=5
visibility=60%
emission angle=90
emission range=70
life variation=8
number variation=6
size variation=6
velocity variation=6
weight variation=2

Size over Life went from 80 to 120%.
Weight Over Life went from 75 to 100%.
Spin over Life went from 30 to 100%.

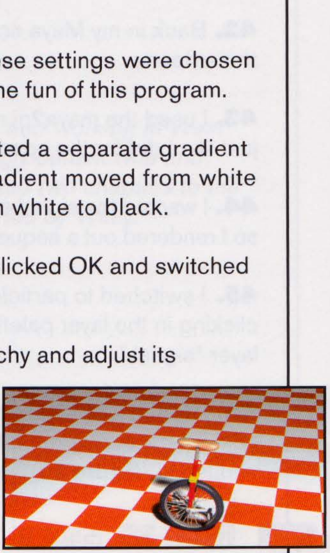
Now things look smokier. Of course, these settings were chosen after much fooling around, but that's half the fun of this program.

39. I switched to the color tab and created a separate gradient for color and transparency. The color gradient moved from white to gray and the transparency moved from white to black.

40. Once I was happy with the results, I clicked OK and switched back to the main stage (**figure 9**).

41. I can still expand smokey in the hierarchy and adjust its attributes a little more.

The animation now has the finger emitting blue sparks and a little smoke.



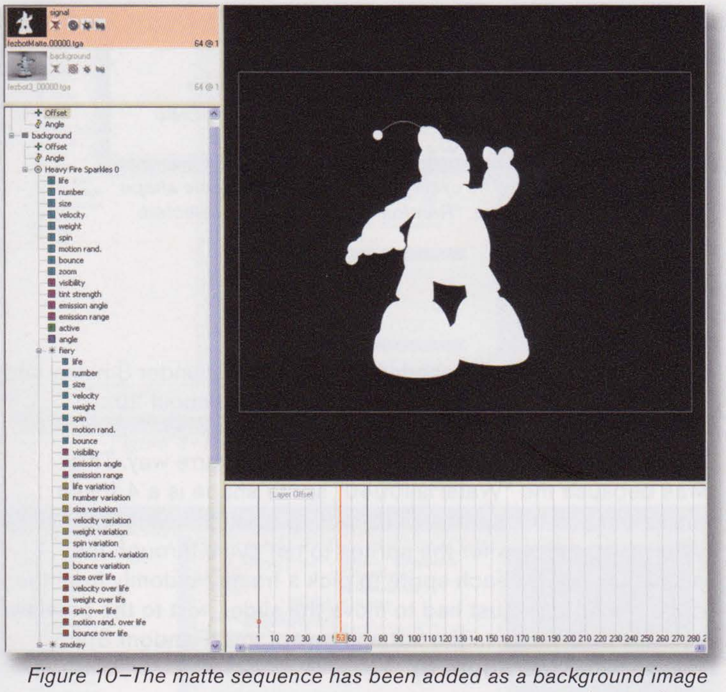

signal
fezbotMatte.00000.tga
64 @ 1
background
fezbot3_00000.tga
64 @ 1
Offset
Angle
background
Offset
Angle
Heavy Fire Sparkles 0
life
number
size
velocity
weight
spin
motion rand.
bounce
zoom
visibility
tint strength
emission angle
emission range
active
angle
fiery
life
number
size
velocity
weight
spin
motion rand.
bounce
visibility
emission angle
emission range
life variation
number variation
size variation
velocity variation
weight variation
spin variation
motion rand. variation
bounce variation
size over life
velocity over life
weight over life
spin over life
motion rand. over life
bounce over life
smokey
Layer Offset

Figure 10–The matte sequence has been added as a background image to the new layer.

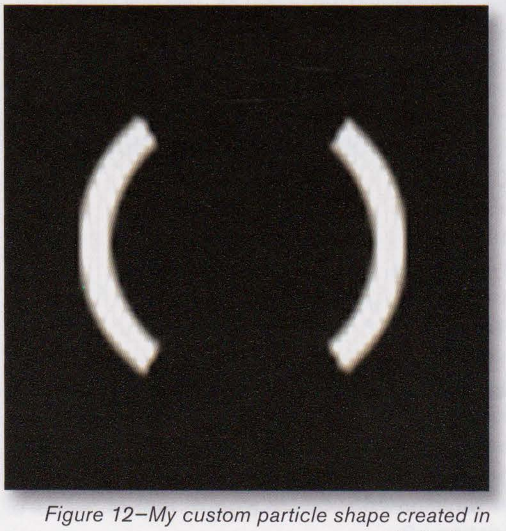


Figure 12–My custom particle shape created in Photoshop.

I wanted to add a third particle to the animation just to show off a few more features in particleIllusion. This third particle will be emitted from the robot's antenna as a sort of warning signal. I guess he's trying to tell someone that his finger is on fire.

42. Back in my Maya scene, I parented a locator to the ball on the antenna.

43. I used the maya2pi.mel script to export the positional data, just like I did for the sparks.

44. I wanted the signal emitter to be occluded by the robot's head, so I rendered out a sequence with just the robot as a white matte.

45. I switched to particleIllusion and created a new layer by right-clicking in the layer palette and choosing "New Layer." I named this layer "signal."

46. I added a background image sequence to this layer, just like I did for the background layer. This time I chose my robot matte sequence (**figure 10**).

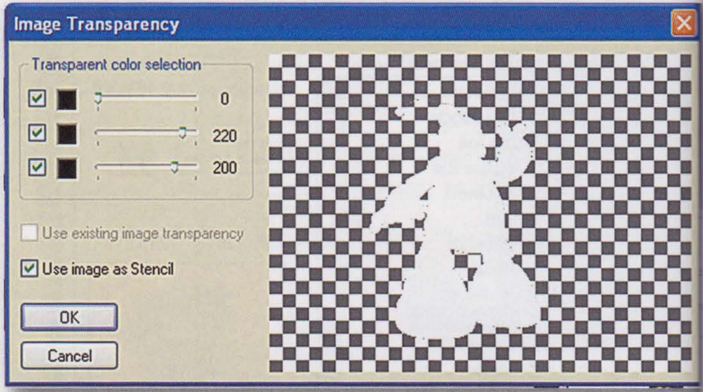
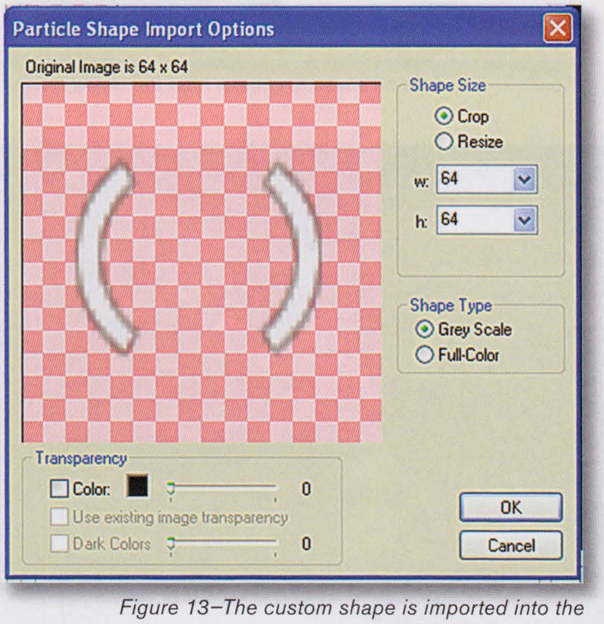


Image Transparency

Transparent color selection
☑ 0
☑ 220
☑ 200

☐ Use existing image transparency
☑ Use image as Stencil

Figure 11–The transparency settings for the new laye



Particle Shape Import Options

Original Image is 64 x 64

Shape Size
● Crop
○ Resize
w: 64
h: 64

Shape Type
● Grey Scale
○ Full-Color

Transparency
☐ Color: 0
☐ Use existing image transparency
☐ Dark Colors 0

Figure 13–The custom shape is imported into the particleIllusion library.

47. In the layer palette I clicked on the "T" icon next to the layer image. This turned transparency on.

48. I right-clicked on the layer and chose "Transparency " from the menu. This brought up a dialogue box that allowed me to set the transparency matte for the layer (**figure 11**).

49. In Photoshop I created a simple black and white 64x64 image for my signal particle (**figure 12**).

50. In the signal layer I added a particle. I just used Simple Sparkles to start with. I imported the positional data from my signal text file that was exported from Maya. The emitter jumped to the antenna and the keyframes were set on the timeline.

51. In the properties box for the simple sparkles emitter, I used the buttons below the "shapes in Library" section to import my custom shape from Photoshop. This added the shape to the library for use in future projects (**figure 13**).

52. I went through the process of adjusting the settings on the attributes until I got the look I liked. I also checked the "attached to emitter" box, which meant that all the particles would follow the movement of the emitter.

FRAME THROWER™

Reliable realtime uncompressed 2K playback

"The Frame Thrower allowed us to manage a huge volume of work from multiple companies and still maintain the quality. Digital Ordnance provided outstanding support. I wouldn't want to make another film without a Frame Thrower."
-Randy Starr, Visual Effects Producer, Chronicles Of Narnia

Frame Throwers have been used in the production of over 100 feature films including:

American Dreamz / Failure to Launch / Take the Lead / Hostel Glory Road / The Pink Panther / The Chronicles of Narnia: The Lion, the Witch and the Wardrobe / Yours, Mine and Ours Flightplan / Fantastic Four / Mr & Mrs Smith / The Longest Yard / xXx: State of the Union / Bewitched / Be Cool / Team America: World Police / Serenity / Zathura / Prison Break Deuce Bigalow: American Gigolo / Elektra / Van Helsing Master and Commander : The Far Side of the World Sky Captian & the World of Tomorrow / Battlestar Galactica

Digitial Ordnance
1724 Rockefeller Lane
Redondo Beach, CA 90278
310. 733 .3001
digitalordnance.com

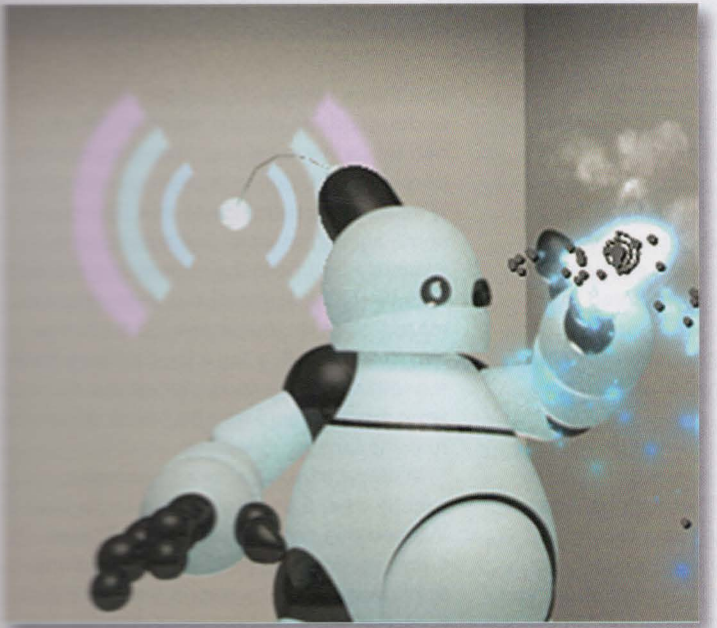


Figure 14–The new particle type swings behind the head of the robot, thanks to the transparency settings on the matte sequence.

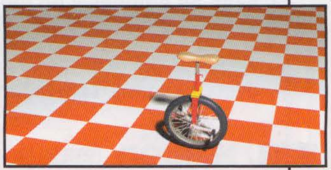
53. Once I clicked "OK," the emitter was added and the antenna is now sending out a warning signal. As the antenna swings behind the head, the signal is occluded, reinforcing the sense of depth in the scene (**figure 14**).

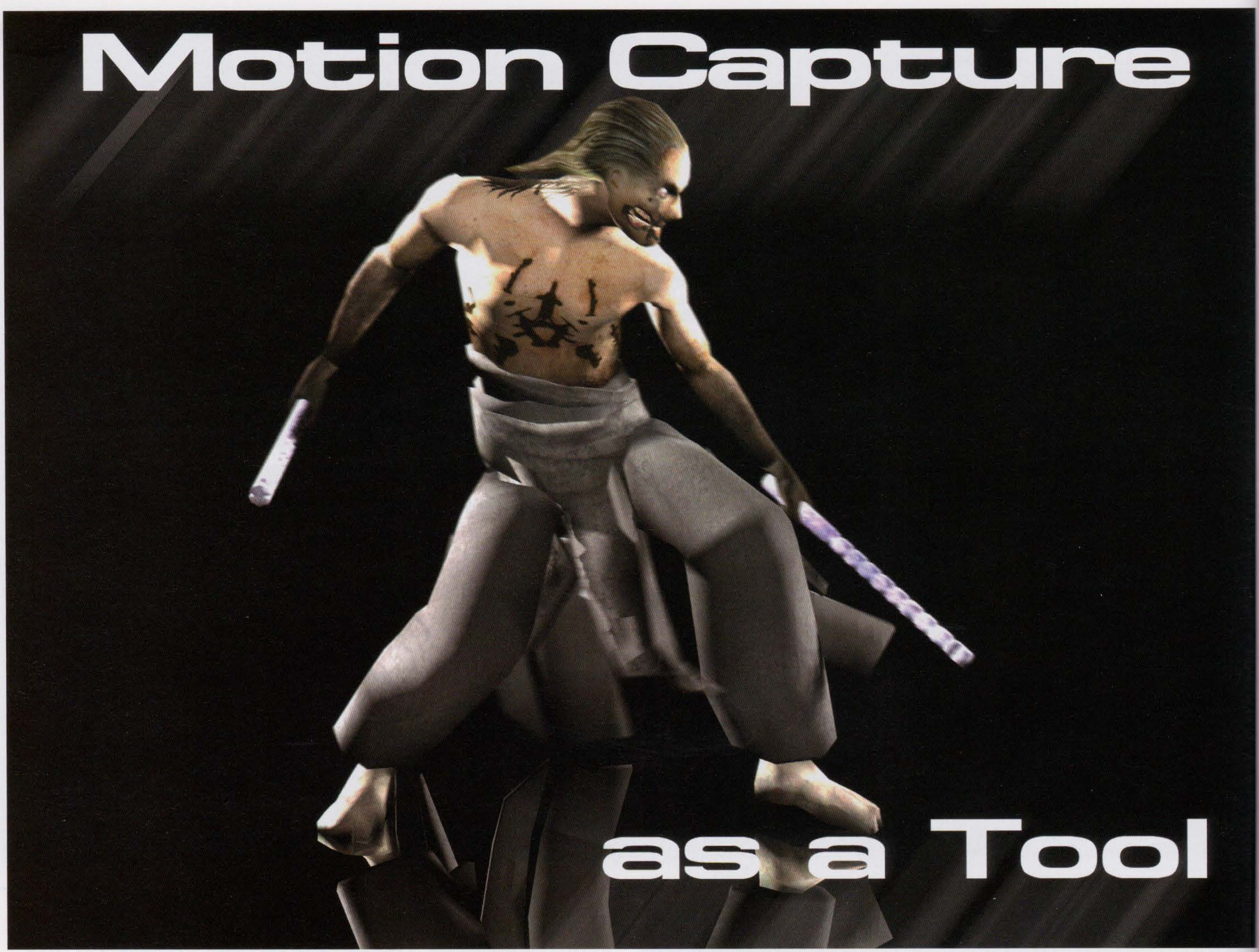
I hope this tutorial has given you a taste for what particleIllusion can do. The QuickTime on the *HDRI3D* Web site shows three versions of the animation before and after the various emitters were added in particleIllusion. You can render out image sequences of the animation directly from particleIllusion, or you can render out just the particles with an alpha channel for compositing in After Effects or another compositing program. Rendering in particleIllusion is much like Maya's hardware render buffer in that it stores a screenshot of each image in the sequence. There is a software rendering module available as well. Motion blur can also be set for the particles.

Download the trial version and play with it a bit. If you use particles a lot in your work, you may find particleIllusion to be a real time-saver for many tasks.

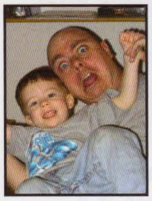
Good Luck!

***Eric Keller** is a freelance animator living and working in West Hollywood, California. He has an amazingly patient wife and two great dogs. He has recently contributed two chapters to the upcoming* Mastering Maya 7 *book published by Wiley.*





Motion Capture as a Tool



By Richard Lico
Character Animator
Bothell, Washington

In the modern gaming industry, Motion Capture (mocap) has become an essential tool in helping to create a representation of realism in animation. Everything from the actions of football players, to space marines firing a rocket launcher, mocap is everywhere. And why not? The mocap look is hot right now. With mocap equipment, it's possible to capture hundreds, even thousands, of motion segments in a matter of days or weeks. Having a smaller animation team using mocap as a starting point not only reduces cost, but it also speeds up the process of creating animation. And the near future looks even brighter for mocap. With an emphasis on movie quality visuals in the next console generation, the need for more realistic characters will become overwhelming. This means creating much more animation per character than in the past, quickly and cheaply, which are exactly the strengths of mocap.

Yet through the use of mocap, the uniqueness of animation itself can be compromised. Granted, you wouldn't use mocap unless you needed a subtle realism look, but does that mean you need to forego the strengths of what mocap offers if you're looking for a more aesthetically varied end result, like hyper realism for example?

When I obtained my current position as senior animator on *Condemned: Criminal Origins* with Monolith Productions, I was told I'd be using mocap "as a tool." So after various phrases filled my head, all ending in the word "suck," I cooled off, and decided to give mocap a chance. While working for previous employers, I had the pleasure of hand animating characters on numerous games, and my limited experience with mocap had been more of a plug-'n'-play where we, as animators, would clean the motion, which consisted of simply making sure the hands and feet interacted properly, and that the motion fit the scene. I was told "this is the way to achieve realism" by people who didn't understand the mocap process. As you can tell by my initial reaction, the general perception of mocap from most animators is less than artistic due to this. There are some animators who view mocap as a removal of art from the animation process entirely. This perspective is assuming that mocap does an animator's job for him or her. Yet Monolith's "as a tool" phrase made me rethink my previous evaluations of what mocap could be and how it relates to realistic looking animation.

Without thinking about it, we know exactly what a person walking looks like. We see this happen hundreds of times a day, so every subtlety is engrained into our heads. Since mocap is a recording of a person's locomotion, logic dictates that mocap applied to a character should look perfect. Yet, in many cases, when raw mocap is applied to a character, it tends to look vaguely stiff and soul-less. One of the main reasons is because it's lacking hyper realism. What is realistic animation anyway? Over the years, I've learned that "realistic animation" isn't what the name implies. Often, when your art director

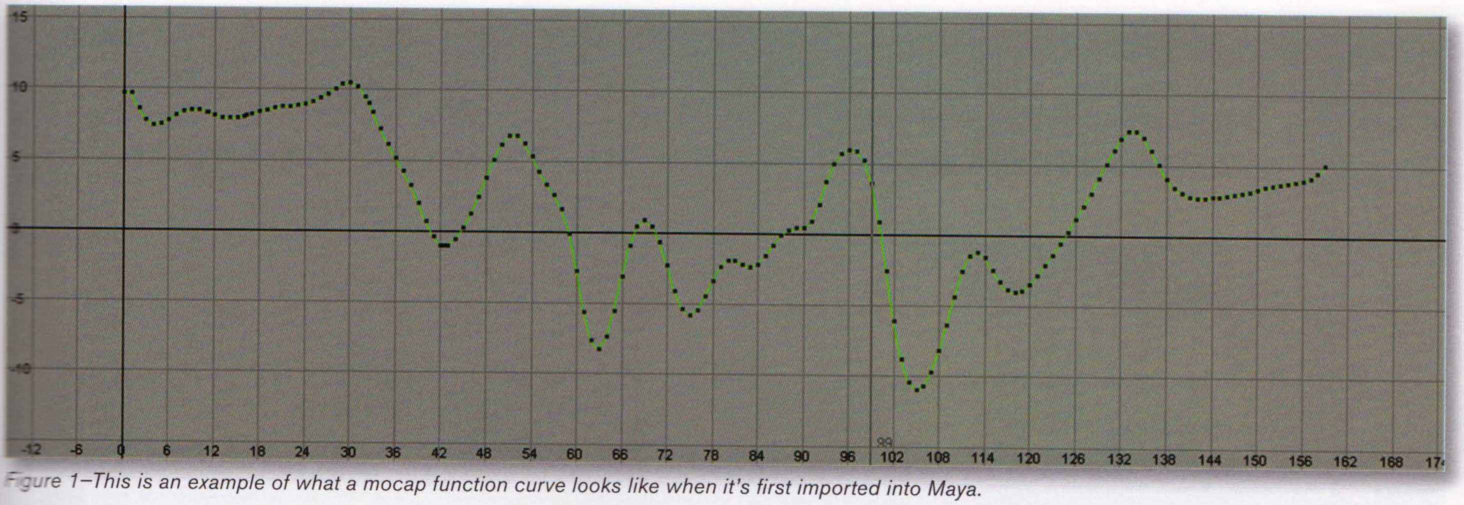


Figure 1–This is an example of what a mocap function curve looks like when it's first imported into Maya.

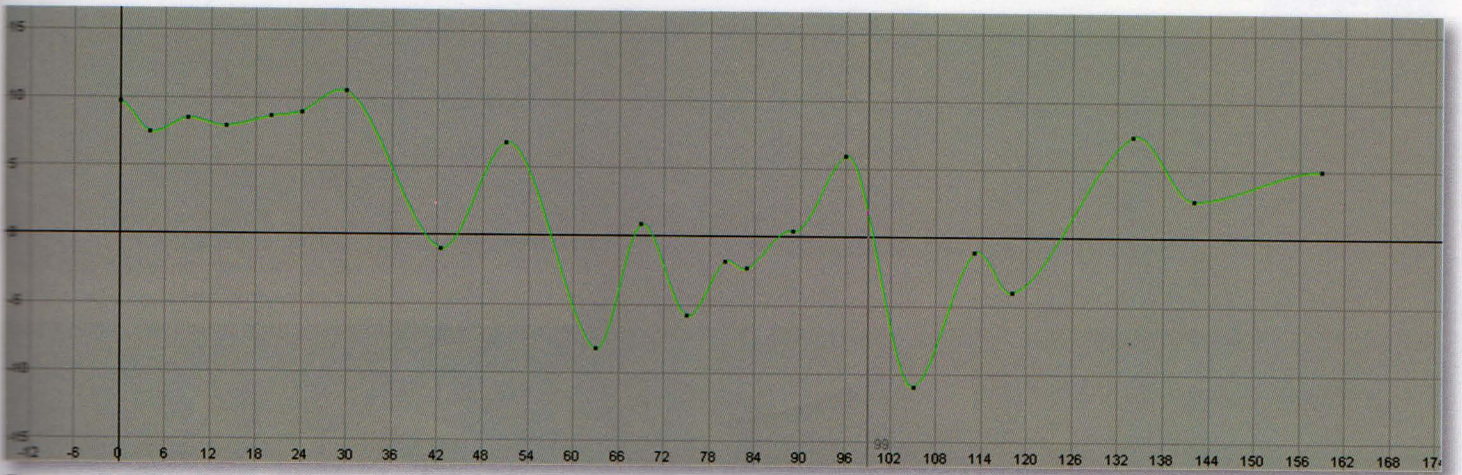
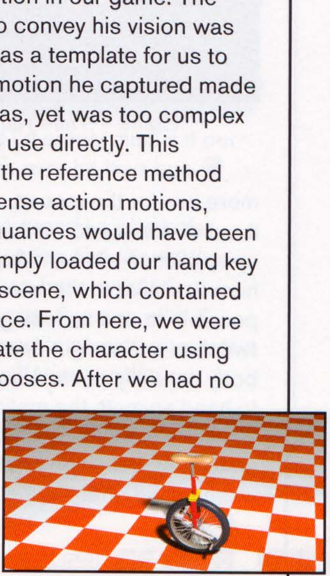


Figure 2–After ridding yourself of the troublesome redundant keys, this is what your function curve should end up looking like.

says he wants "realism," what he really wants is "hyper realism." Hyper realistic motion is essentially an idealized version of realistic motion. When you see a character animated well, you don't think about the motion, you simply accept the action. When an animation hasn't been done well, you tend to pay more attention to what's wrong, and less to the overall action. What makes an animation more believable, is an extension of what's possible. The goal is to go beyond what an actor can do, without entering the realm of being unbelievable. It's to attain a sense of exaggeration of perfection.

So how do we add hyper realism using mocap as a tool? There are a few different ways to approach this. The most obvious is to simply edit the mocap. By selecting a bone, you can see its f-curves in your graph editor. (The names of the windows I'm referring to are from Maya.) You'll notice that there are keys on every frame within your curves (**figure 1**). To be able to edit the motion correctly, you'll need to break those keys down into the essential motion. Begin by deleting the redundant keys, leaving only the ones necessary to define the function curve's shape (**figure 2**). I've found that keeping the high and low peaks of the curves works best, but this rule is not a catch-all. Having a critical eye and understanding animation is the most valuable way to judge what should stay and what should go. Depending on your 3D package of choice, you should be able to find user written scripts to automatically complete this step for you. Just a warning though: A script doesn't have the aesthetic eye you may have, which at times leaves you with less than stellar results. Once you have only the necessary keys, begin to edit your animation as you would if it were hand keyed. To be honest, this method has been known to cause animators to mumble all evil-like, and involuntarily twitch. It's a time consuming, tedious method that ends up removing much of the subtle nuances that make mocap valuable in the first place. The only advantage I see in using this method is having the ability to edit the motion after the key reduction, as if it were hand keyed.

Another method, which we used from time to time on *Condemned*, is to reference mocap. Mocap makes a much better reference than a video file. You can view the motion frame by frame, at any angle, while you work in your scene. It also saves in the planning, timing, and staging process compared to non-referenced hand key animation. With *Condemned*, our lead designer had a target look he wanted to achieve for the motion in our game. The way he was able to convey his vision was to capture mocap as a template for us to see. Some of the motion he captured made for some great ideas, yet was too complex in certain cases to use directly. This enabled us to use the reference method for a few of the intense action motions, where the subtle nuances would have been overlooked. We simply loaded our hand key character into the scene, which contained our mocap reference. From here, we were able to hand animate the character using similar timing and poses. After we had no



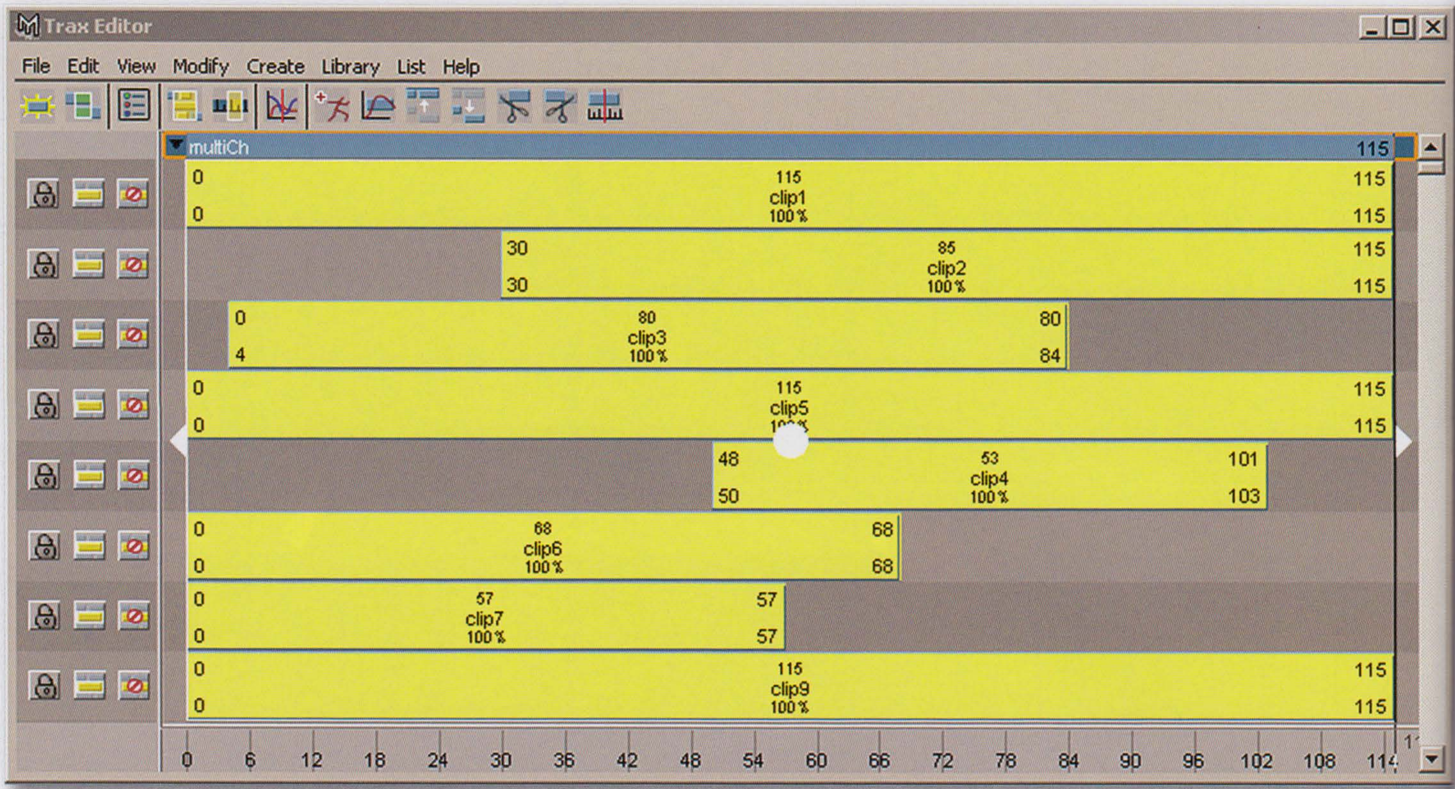

Trax Editor
File Edit View Modify Create Library List Help
multiCh 115
0 115 clip1 100% 115
0 115
30 85 clip2 100% 115
30 115
0 80 clip3 100% 80
4 84
0 115 clip5 115
0 115
48 53 clip4 100% 101
50 103
0 68 clip6 100% 68
0 68
0 57 clip7 100% 57
0 57
0 115 clip9 100% 115
0 115
0 6 12 18 24 30 36 42 48 54 60 66 72 78 84 90 96 102 108 114

Figure 3—Layers are your friends. Don't be afraid to use as many as you nee

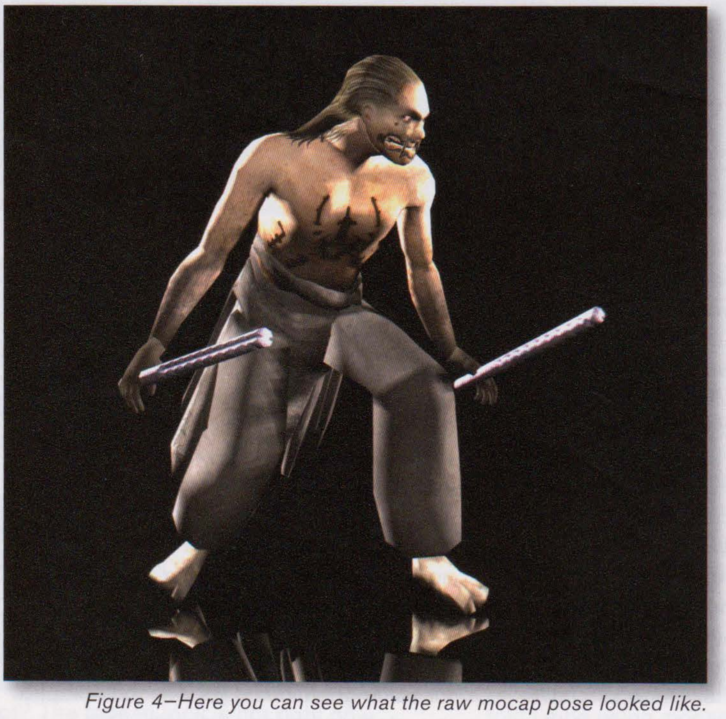


Figure 4—Here you can see what the raw mocap pose looked like.

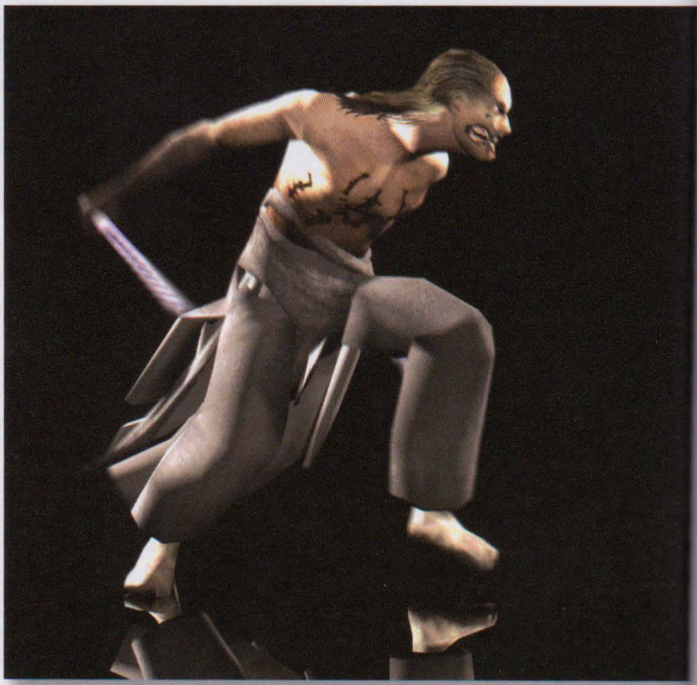


Figure 5—And here's the same pose after layers of animation on to

more use for the mocap, we would delete it out of our scene, and continue adding spit-polish to our hand key. The downside here is similar to the one above. It's possible to end up losing a lot of the subtle twitches and weight compensation the body naturally does. Also, the time it takes to hand animate the motion is pretty steep. So with the short development cycle, and amount of animation needed, this method wasn't used often.

The final method I intend to discuss here involves layering animation on top of mocap. Programs like MotionBuilder have been using it with mocap editing for years. I became aware of this working style during my many years with Character Studio, the 3DS Max character animation plugin. Layering animation was one of its best features, in my opinion. Animation layers work much like layers in Photoshop. You could put your blocked out motion on the bottom layer, and begin pushing your poses, adding overlapping and secondar motion, with additional layers above your blocking. This method seemed perfect for animating on mocap. Since Monolith uses Maya, I found the "Trax editor" to do the trick. (**figure 3**) First I'd make a layer (which Maya calls a clip) out of my raw mocap. Think of it as your foundation, which you'll be building onto. Next, I'd ad a layer above the mocap foundation. In

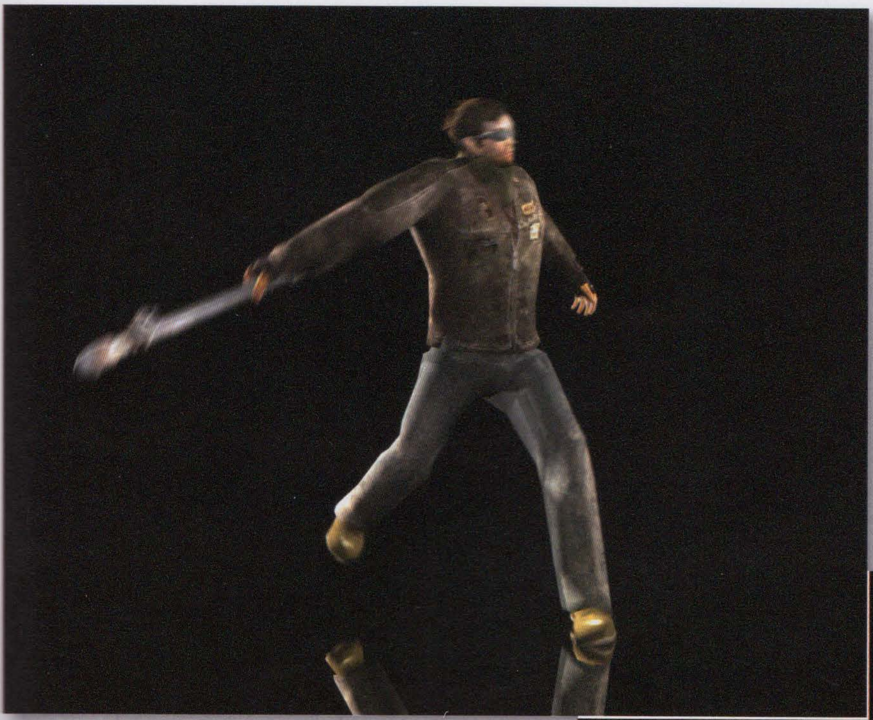
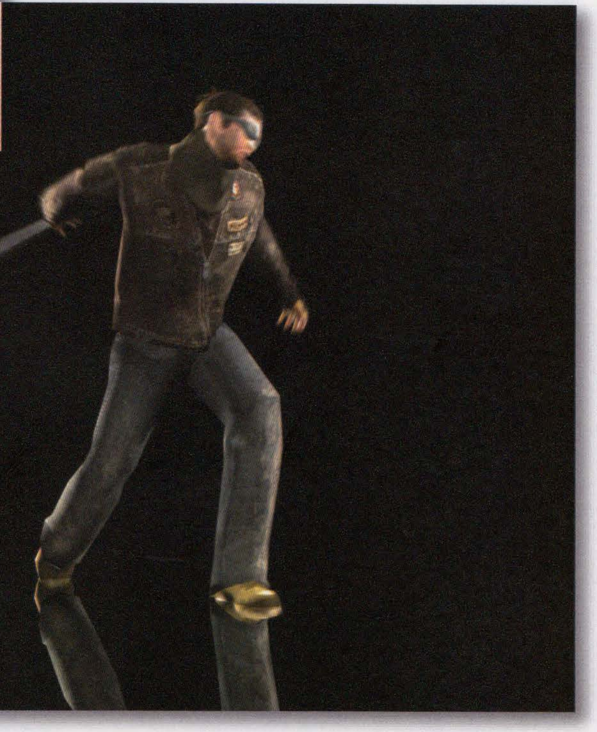


Figure 6 (At left)–This is what my character looked like as straight mocap data was applied. Notice the weak pose, and lack of power in his right shoulder and spine.

Figure 7 (Below)–This is after numerous layers in which I reposed the full figure and used my offset method to add a delay to the arm's swing. Notice the improved drive in the spine and right shoulder. In this case, it not only adds anticipation to the swing, but it also makes the pipe seem like it has more weight.

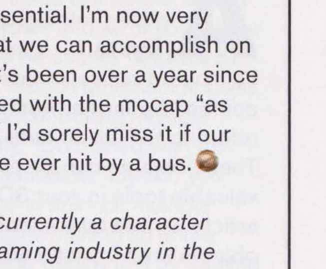


this layer, I'd identify the key poses within the motion, and repose them to be more dynamic and stylized. View each pose that you are redefining as an illustration, or a still shot of perfection (**figures 4** and **5**). Since layers are an offset from the motion in the layer below, I can return to its default motion at any time by setting keys with an offset value of zero. Layers also enable me to retain the subtlety and nuances of the motion, and still have the ability to hand animate on top.

Next, by adding more layers, I'd edit the overlapping and follow through motion. This step is difficult to explain, so I'll do so through an example. Let's say your mocap character's arm is swinging a lead pipe over the duration of about eight frames. A way to make it look and feel more powerful would be to add more time to the anticipation and follow through, and less to the actual swing. In a new layer, set a zero key for the arm on the frame when it is fully extended during the swing. Since it's a key with zero offset, nothing yet has changed. Two frames before the extension, repose the arm to look as if it has not yet begun to swing. By doing this, you remove some of the time it takes to swing, while adding to the anticipation time (**figures 6** and **7**). This may cause the original starting pose for the swing to break. If this occurs, simply set a key on the arm with an offset of zero on the frame in which it's the most busted looking. Use this method to shorten the latter half of the swing as well. The end result will be a more powerful looking swing. Once I was happy with the overall look of the motion, I'd collapse all my layers and apply the motion back onto the character. By time scaling chunks of the keys to further accentuate the anticipation and follow through, I'd change the timing of the motion to better fit the new, dynamic style. Afterwards, simply clean up the foot sliding that's naturally involved with mocap work (if needed). If you're interested in viewing before and after example footage, you can download an assortment of my Maya playblasts from the Resources page at *www.hdri3d.com.*

Now that mocap is a much more essential tool in the animation workflow for reality based games, it's our job as animators to push the boundaries beyond what mocap inherently is. I'm not saying that plug-'n'-play mocap doesn't have value in some cases, but I am saying that we're not limited to sticking with one aesthetic result. In the case of *Condemned*, having an animation style individual from the competition, yet still semi-realistic and appealing, was essential. I'm now very excited to see what we can accomplish on our next project! It's been over a year since I was first presented with the mocap "as a tool" theory, and I'd sorely miss it if our mocap studio were ever hit by a bus.

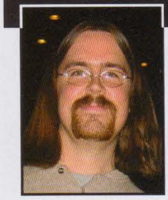
***Richard Lico** is currently a character animator for the gaming industry in the Seattle area. He has been animating professionally for over 6 years, with 8 published game credits as well as various broadcast and illustration credits.*



GRAB LIGHTWAVE BY THE NULLS!

An In-Depth Look at LightWave's Null Objects

Figure 1–Illustration by Alejandro Parrilla, model by Proton.



William "Proton" Vaughan
3D Artist, Author, Teacher
Orlando, Florida

"The cause is hidden. The effect is visible to all." —Ovid

A Null is as that which has no value (nothingness) or no effect (**figure 1**). It originates from the Latin word nullus, meaning not any. This couldn't be farther from the truth when referring to Null objects in LightWave. These objects can be one of the most valuable tools in your 3D toolkit. In this article we will take a look at why Null objects are so useful and explore some of the many uses for them in a LightWave production. So, what is a Null anyway?

LIGHTWAVE NULLS DEFINED

I recently gave a lecture about managing large scenes in LightWave, and a discussion about Nulls started. One of my students asked, "So what is a Null anyway?" and I had to stop and think of the best way to define a Null object. Here's the simplest way I can explain what a Null is.

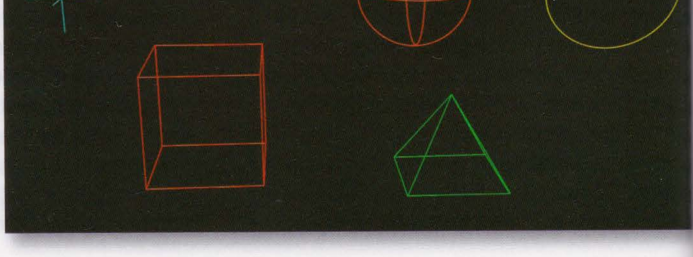
Null objects are special objects in LightWave Layout that serve as reference points in 3D space. By default, Null objects appear as six-pointed stars or jacks, but you can change the appearance of a Null object by applying "Item Shape" in the Add Custom Object list. They can be moved, rotated, scaled, and so on, just like any regular object (**figure 2**).

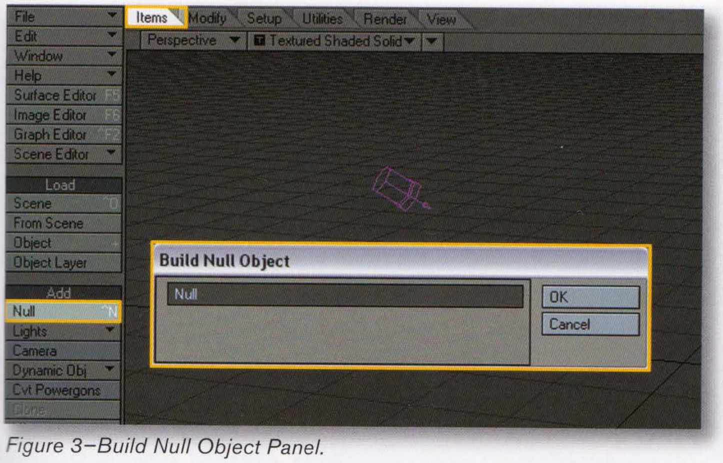
It is important to remember that though they have a lot of the same attributes of a LightWave object, they have some unique features that set them apart. Null objects are created by LightWave on a scene-by-scene basis and are not real objects that can be saved to disk. They remain part of the scene file, however, and will reload with the scene.

Null objects, like the camera and lights, will appear to grow or shrink in relation to objects as the grid size is changed. Null objects contain no geometry so they never render in an image. Although they never render in an image, they can affect render times, as noted by Dave Jerrard in *LightWave 3D [8]: 1001 Tips & Tricks* by Wordware Publishing (*http://www.wordware.com/*).

Dave Jerrard writes: "Even though Null objects are not visible in a render, they can add drastically to render times when Ray

Figure 2–Examples of Null Objects. The Null on the far left has the default shape.




File
Edit
Window
Help
Surface Editor
Image Editor
Graph Editor
Scene Editor
Load
Scene
From Scene
Object
Object Layer
Add
Null
Lights
Camera
Dynamic Obj
Cvt Powergons
Items
Modify
Setup
Utilities
Render
View
Perspective
Textured Shaded Solid
OK
Cancel

Build Null Object

Null

Figure 3–Build Null Object Panel.

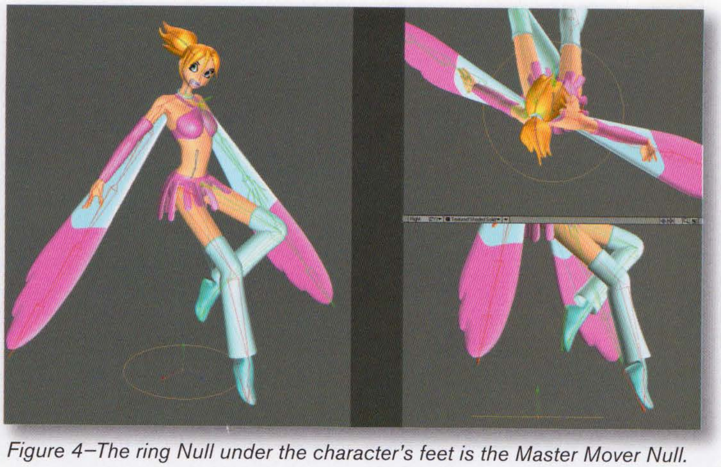


Figure 4–The ring Null under the character's feet is the Master Mover Null.

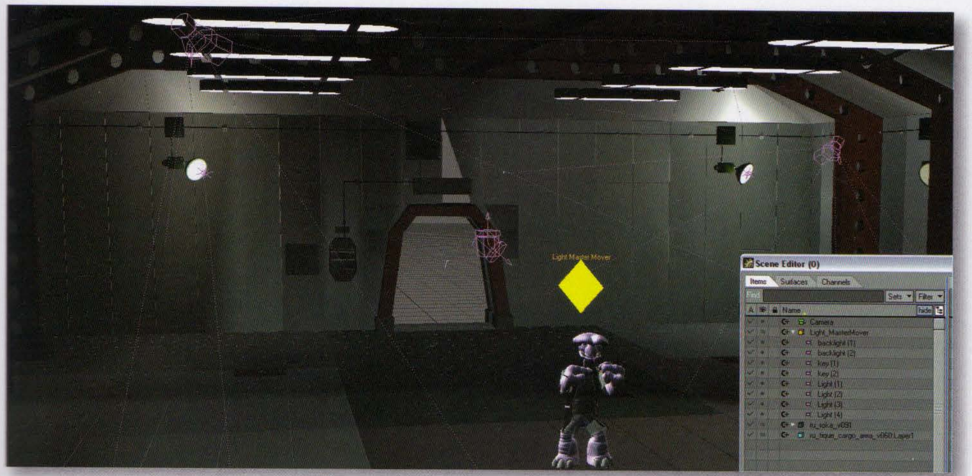

Light Master Mover
Scene Editor
Items
Surfaces
Channels
Camera
Light_MasterMover

Figure 5–The yellow Null object above the character's head is the Master Mover for all the lights in the Scene. All the Lights are parented to this Null.

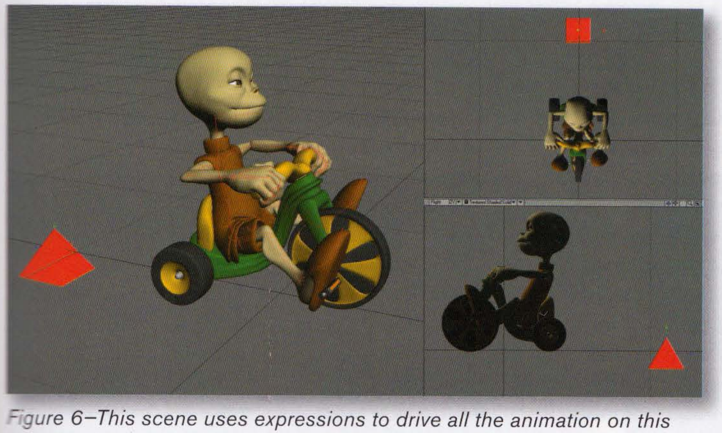


Figure 6–This scene uses expressions to drive all the animation on this big wheel setup based on the movement of the red Null object.

Trace options are used. Internally, LightWave still checks each one to see if it could cast or receive a shadow, and then does the ray tracing function later. It's this check that adds the time. In scenes with a lot of Nulls (50 or more), this can add up to five minutes per frame. By making each Null unseen by rays, and unchecking the shadow options for it, you take the Null out of that check routine. It also doesn't hurt to disable the Null in the Scene Editor by unchecking it as well. This removes the item from all rendering considerations, but doesn't change how it might affect motion or displacement of other objects." Dave will be happy to know that this has been fixed in LightWave 9, which should save all of us quite some time.

Now you know what a Null object is, but how do you add them to a scene?

CREATING A NULL

Null objects are created in LightWave Layout by following three simple steps:

STEP 1: Select *Items > Add > Null* or use the keyboard shortcut (Ctrl+N).

STEP 2: In the text field type in a name for your Null object. The default name is simply "Null."

STEP 3: Click OK.

There are several tools in LightWave Layout that generate Null objects, but 95% of the time you will use the steps above (**figure 3**). The workflow for Null creation has been improved in LightWave 9, and a demonstration of this can be found at *ftp://ftp.newtek.com/pub/LightWave/LW9/BuildNull.mov.*

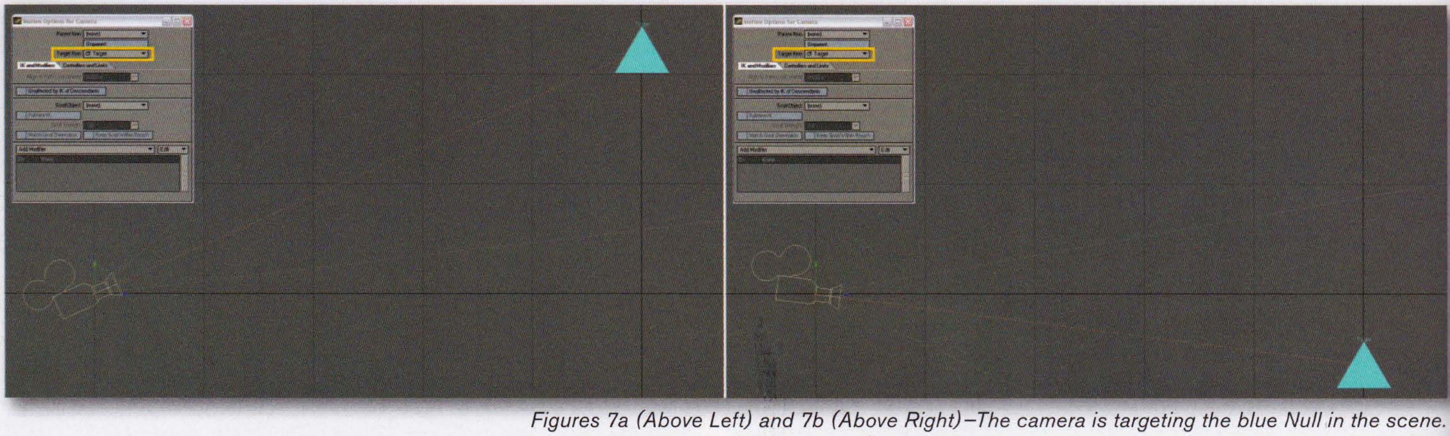
Now you have a Null object. So, what can it be used for?

ONE NULL TO RULE THE ALL

One of the most common uses for Null objects is as a "Master Mover." Parenting a hierarchy of items to a single Null object usually referred to as a "Master Mover" helps group components in a scene. "Don't underestimate the uses for Null objects," notes Lee Stringer, Instructor at the DAVE School. "I always use Null objects to help organize my scenes and assemblies and find it difficult to navigate a scene that doesn't have this grouping of items" (**figure 4**).

Light rigs are a great example of using a Null as a "Master Mover." By parenting all of your lights in a scene to a Null object, you can relocate the entire light "rig" by simply moving/rotating the Null. This can shave massive amounts of time off your workload. Kevin Nations, Director of UI Design at NewTek, offers this helpful tip: "If you need to scale a group of items in a scene, simply scale the Master Mover and all of your items in that hierarchy will be scaled" (**figure 5**).

Expressions and Motion Modifiers applied to Nulls are great for controlling multiple items simultaneously. "Combined with the power of the motion modifier 'Cycler,' you can use a single Null to animate multiple items at a time. Imagine making an entire landing gear assembly deploy just by rotating a single Null!" adds Chris Nelson, Instructor at the DAVE School (**figure 6**).



Figures 7a (Above Left) and 7b (Above Right)–The camera is targeting the blue Null in the scene.

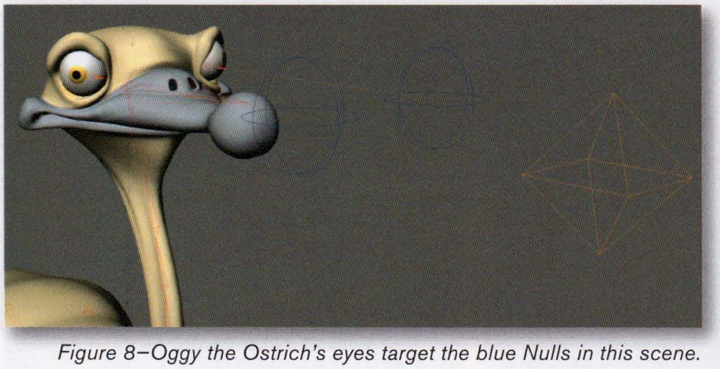


Figure 8–Oggy the Ostrich's eyes target the blue Nulls in this scene.

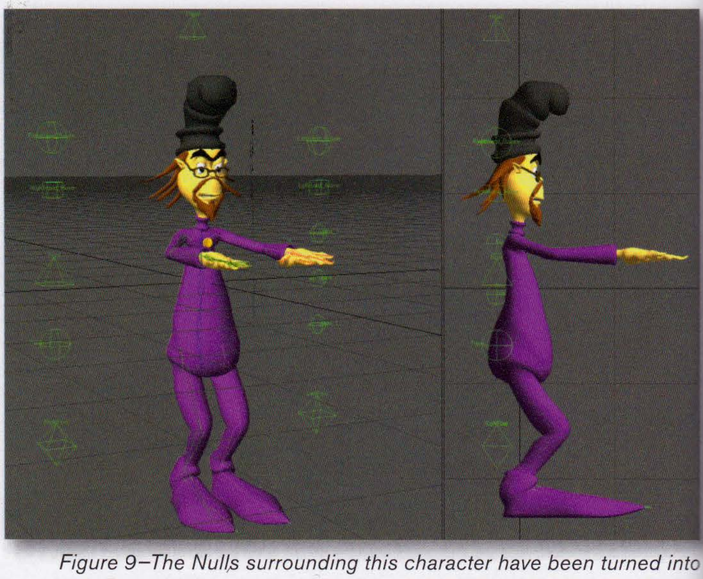


Figure 9–The Nulls surrounding this character have been turned into Proxy Items to make selecting the bones in the character easier.

STAY ON TARGET...STAY ON TARGET!

Nulls can act as reference points for objects, lights, or cameras to watch or track. "The beauty of using Null objects as targets is that they don't show up in the final rendered image even if they pass right in front of the camera," notes Dan Dod, Animator at UFO Films (**figures 7a** and **7b**).

To have an item track a Null's movement, follow these easy steps:

STEP 1: Select the Item that you want to track the Null.

STEP 2: Choose *Windows > Motion Options.*

STEP 3: From the Target Item drop-down list, choose the Null object you wish to target.

A couple years ago I wrote a tutorial that shows how to setup an eye rig using Nulls that the eyes of a character target to speed up character animation and help maintain eye-lines. The tutorial can be found at *http://www.newtek.com/products/lightwave/tutorials/animation/eyes/index.html* (**figure 8**).

GOAL OF THE NULL

Nulls are commonly used in the rigging process to act as goals for Inverse Kinematics and to control hierarchies. My first real use of Null objects was when I ventured into rigging characters and needed to create goal items for my IK chains. I later used Null objects as Proxy objects and control handles for more advanced rigs. Deuce Bennet, LightWave Beta Coordinator, created an introduction to using Nulls as Proxy objects that can be found at the NewTek site at *http://www.newtek.com/products/lightwave/tutorials/animation/proxy/index.html* (**figure 9**).

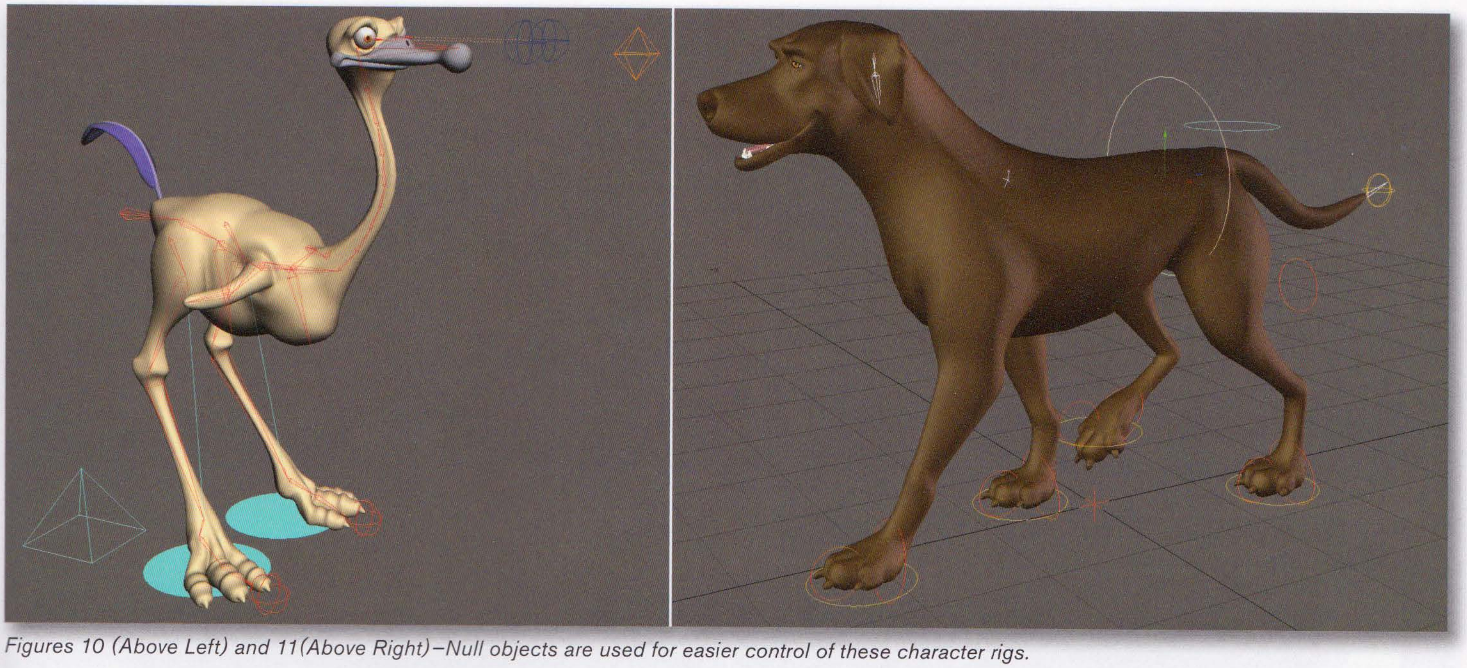
"Nulls give me the power to control my characters in virtually unlimited ways. That level of control is what animators need to get the job done and make deadline," states Chris Nelson, DAVE School Instructor (**figure 10** and **11**).

Null objects can also be used to rig a vehicle to achieve sliding, tilting, and banking effects to emulate realistic movements. Alejandro Parrilla demonstrates this in an easy-to-follow tutorial located on NewTek's site at *http://www.newtek.com/products/lightwave/tutorials/animation/car_setup/index.html* (**figure 12**).

PIMP MY NULL

As mentioned earlier in this article, you can change the appearance of a Null object by applying "Item Shape" in the Add Custom Object list located in the Object Properties panel. Item Shape lets you specify the shape and look of Null objects and is a great way to give each Null in your scene a unique identity. "The ability to customize the look of Nulls comes in handy when dealing with complex scenes that have multiple Nulls," remarks Ivan Young, DAVE School Instructor. "I can quickly find the Null I need to select at any given time."

You can also turn your Null objects into handy measuring devices using custom objects. Add the Ruler custom object to add a virtual ruler to your scene (**figure 13**), or the Protractor custom object to measure angles (**figure 14**). Have your Null object measure the speed of an item in meters per second by adding the Speedometer custom object (**figure 15**).



Figures 10 (Above Left) and 11(Above Right)–Null objects are used for easier control of these character rigs.

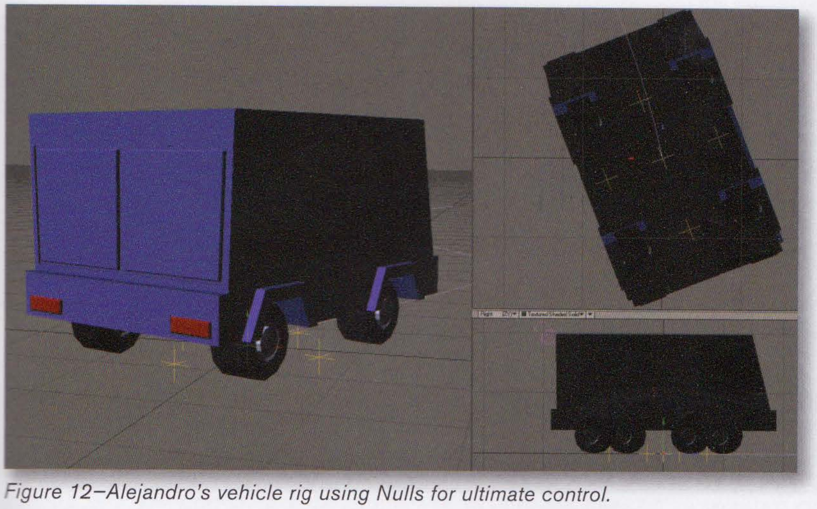


Figure 12–Alejandro's vehicle rig using Nulls for ultimate control.

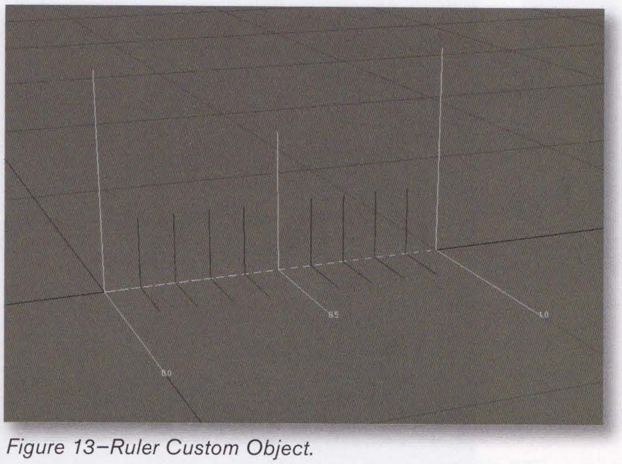


Figure 13–Ruler Custom Object.

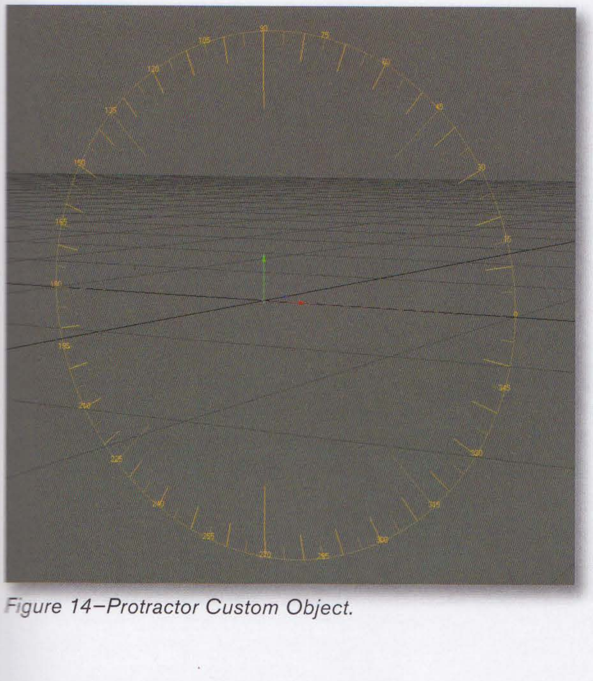


Figure 14–Protractor Custom Object.

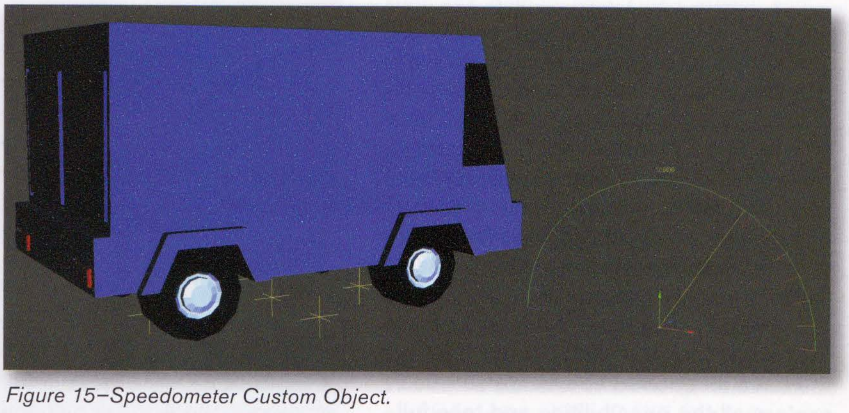
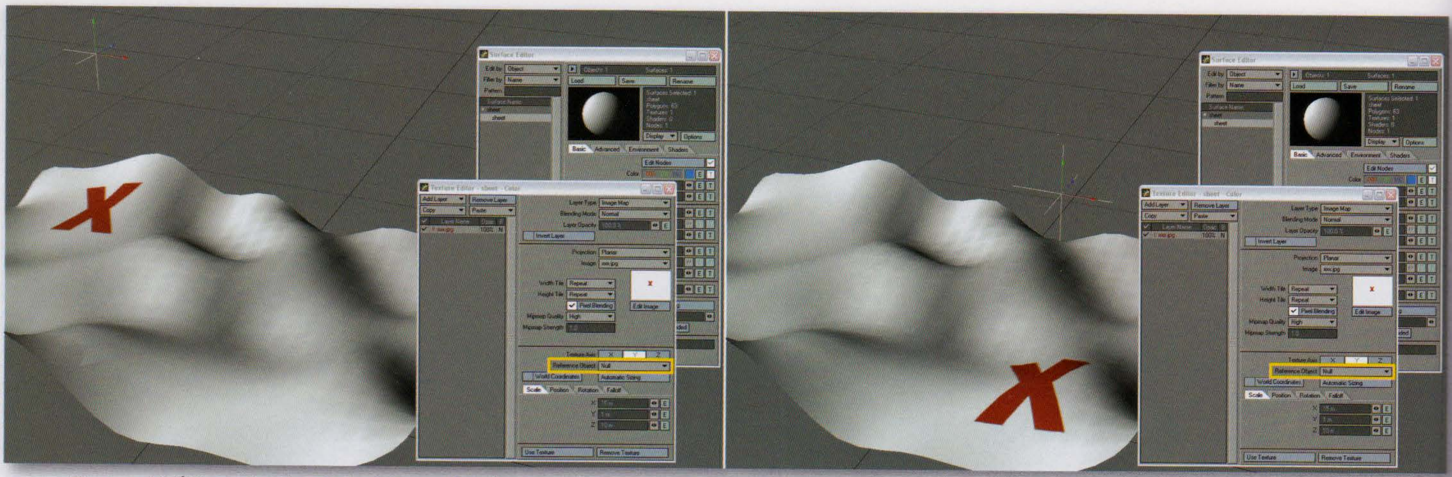
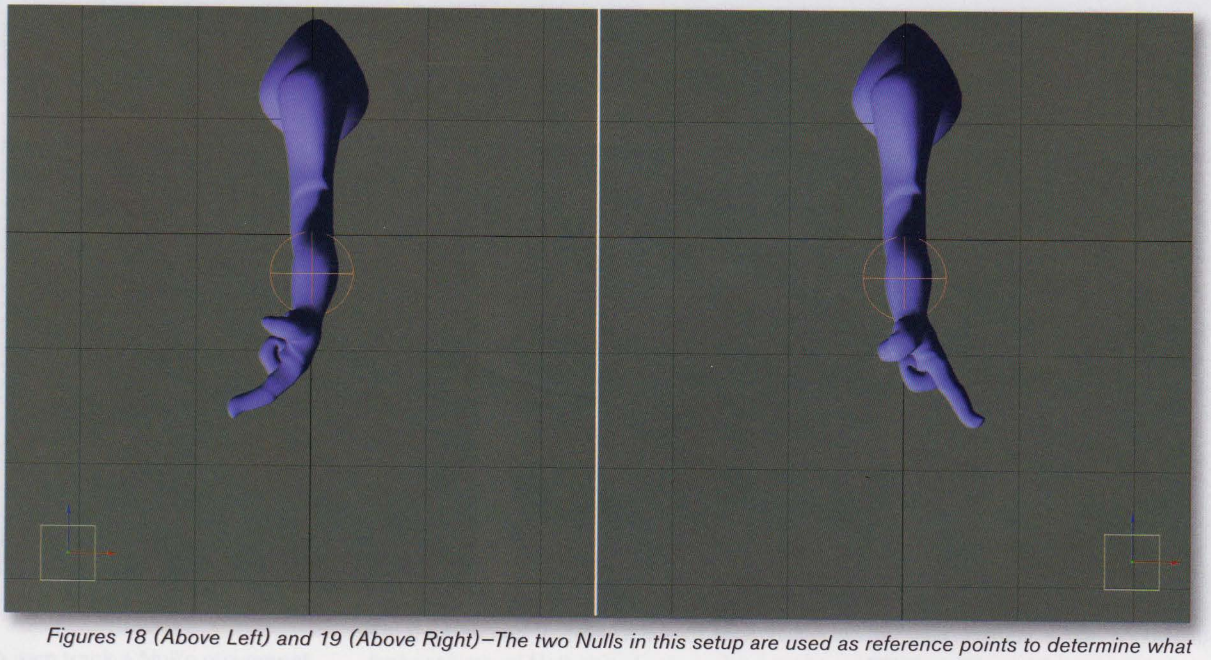


Figure 15–Speedometer Custom Object.



Figures 16 (Above Left) and 17 (Above Right)–The Null in this setup is used as a Reference Object to control the placement of the X texture that is applied to the white object.



Figures 18 (Above Left) and 19 (Above Right)–The two Nulls in this setup are used as reference points to determine what portion of the chosen mesh will be affected.

Adding Range Finder to an object will display the distance in meters from the selected item in the scene. "I find using a Null with the Range Finder custom object very useful in determining the camera's distance to an object to adjust the camera properties appropriately," notes Doug Brown, Animator at AC Coin.

There are many more ways to customize your Nulls in addition to the ones mentioned above. Be sure to explore all the possibilities and take full advantage of these powerful options.

REFERENCING A NULL

Using a Null as a Reference Object in the Texture Editor gives you a way of controlling a texture's position, rotation, and scale on a surface. "I have found that using a Null as a Reference Object can be one of the fastest ways to place a texture in the exact location I want it," notes Ben Vost, NewTek Europe. You can access a couple of video tutorials I created showcasing this technique at the NewTek site at *ftp://ftp.newtek.com/pub/LightWave/Tutorials/Vidz/RefObject_web.mov* and *ftp://ftp.newtek.com/pub/LightWave/Tutorials/Vidz/electric_CD.mov* (**figures 16** and **17**).

Many of the displacement tools located in the Add Displacement drop-down menu need Null objects to operate. The Deform displacement tools let you deform objects much like Modeler's deform tools. Each of the Deform tools uses two Null objects as reference points to determine what portion of the chosen mesh will be affected (**figures 18** and **19**).

INFINITE POSSIBILITIES

We've only scratched the surface on how you can put Null objects to use, but I hope you can start to see how powerful these special objects truly are. The possibilities for using Nulls are infinite, and I always get excited when I discover a new use for them. Good luck with your next project. I'd love to hear from you anytime you have a new and creative use for Nulls. Now go out there and grab LightWave by the Nulls!

> *"All we know is still infinitely less than all that remains unknown."*
> *—William Harvey*

XSI base.com

Softimage XSI Community

Forums
Jobs
Interviews
Plugins
Shaders
Contests

www.xsibase.com

ButterflyNetRender™

Leading Network Rendering Manager

77 Studios and over 600 Animators and Artists use it!

Find out why!
Download
your Free Trial:

http://hdri.butterflynetrender.com

FAT BOX FILMS
MeniTHINGS PRODUCTIONS LLC
12f editorial ▸ animation ▸ creative
JBTV
THE EMBASSY
GLUE WORKS animation
APPLIED ART & TECHNOLOGY
THE FILM FOUNDRY
GIGAPIX STUDIOS
ideografics infografia 3D
INNOMATIONS MAKING REALITY OBSOLETE
CREATIVE animation studios
The Pixel Farm
CANNY LADS ANIMATION www.cannylads.com
IGT

NULL TIPS

1 "Animate on a Null and then parent your object/camera/light to it. If you need to change the object for a new revision, the animation is already there." –Lee Stringer, DAVE School Instructor

2 "If I have a camera attached to an object, but the object is not needed in a render pass, I replace the object with a Null so I can keep my camera movement." –Doug Brown, Animator at AC Coin

3 "Use Nulls to mark foot positions to help measure a character's stride length. They are also great for marking a character's destination on the ground. This is similar to the way one would mark the floor of a set with gaffer's tape where actors need to move to, or position themselves, in a shot." –Chris Nelson, DAVE School Instructor

4 "You can even use a Null as an animation aid. For example: Creating a [sweeping] orbiting motion on a camera or object can be extremely difficult to key by hand. Simply by parenting the orbiting item to a Null instantly gives you the power to get perfectly circular motion just by animating the Null's Heading!" –Chris Nelson, DAVE School Instructor

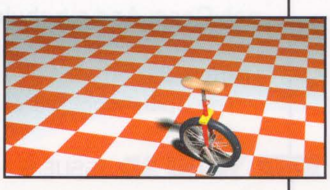
5 "Sometimes I tell my students to work out a particularly complex character motion by blocking out the timing first with a Null. Once they have the rough timing worked out on an action, it is easy to use the keys as visual cues for animating their characters." –Chris Nelson, DAVE School Instructor

6 "When you're setting up an object that will use Nulls for controls, such as IK goals or Reference Objects, make those Nulls part of the file itself. By placing a single point in an objects layer, you turn that layer into a Null object. Since that Null is part of the object itself, it will be difficult to accidentally lose it. This will make the object and its controls much more transportable, especially in terms of texture Reference Objects." –Dave Jerrard, *LightWave 3D [8]: 1001 Tips & Tricks*

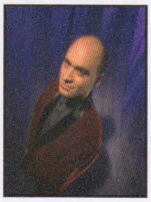
A recipient of several New Media Addy awards, ***William Vaughan*** *has an extensive background in creative 3D for print, web, multimedia, games, and broadcast. During the last 10 years, he has delivered award-winning work for clients such as Compaq, New Line Cinema, and Halliburton. William has also trained artists at several studios and schools around the world and contributed to six LightWave 3D books throughout 2003 and 2004.*

In 2002, Vaughan joined NewTek's marketing team as the LightWave 3D Evangelist, working closely with the LightWave development team, key accounts, and the growing number of end users to enhance LightWave's features set.

William is Director of Industry Relations and Instructor of The DAVE School's upcoming Final Project. William's focus is on continuously improving the quality of education at The DAVE School, while further establishing the school's presence in the industry.



MEL SPLASHFLOCK



Thatcher Bell
Freelance Animator
St. Louis, Missouri

It was a dark and stormy night; the rain tapped on the windowpanes like a hypoglycemic animator at the door of the pizzeria six and a quarter minutes after closing. The phone rang. It was my next freelance job–a 30-second TV ad featuring Dolphins and Macaw parrots in a Cirque du Soleil-esque kaleidoscope of vibrant digital chroma. Among the challenges set before the crew, the "Splash Flock," a burst of Macaws emitting from a dolphin's splash as it reenters the water after an aerial move. After a moment's reflection, I surmised one prudent course of action–mighty, yet simple, MEL scripts. The "Simple Crowd System" chapter of MEL Scripting for Maya Animators by Mark R. Wilkins and Chris Kazmier had the tools I would need to accomplish this task.

I took advantage of the few days before the first production meeting to begin developing this approach to the Splash Flock problem. My preferred scripting tool is UltraEdit, an inexpensive and feature-packed programming application. I extracted the parts I would need from the Simple Crowd System and added a few features specific to my assignment. I used polygonal cones as stand-ins for the Macaws. The cones gave an easy-to-read indication of the motion and orientation of each of the flock elements. The simple geometry also plays back at full frame rate without stuttering in the interface windows.

With an early version script iconized onto the shelf, I was ready for the first production meeting. The production team and management went through the concept and storyboard. When the subject came to the Splash Flock, I suggested a MEL script. There was some initial incertitude about the idea. I gave a brief description of how it would work, then clicked the shelf icon and hit "play." A small group of cones appeared at the origin and flew up and out of the frame. I reinforced my position by estimating a much longer production timeline for hand keyframing the same action. I also promoted the flexibility of the script in that I could change the number of flock elements in a few seconds, giving the Creative Director leeway to make substantial changes without overburdening the production schedule.

Now, on to the scripting. The Splashflock script initializes the number of flock elements, in this example Macaws, and the distance between each bird. It then creates a polyCone, duplicates the polyCone in incremented rotation steps, and distributes them in a circle with each polyCone pointing out from the center (**figure 1**). A Dynamic Solver is created for the scene. Active Rigid Bodies and Radial Fields are created for each polyCone. A "random wander" and "orientation based on motion" expression is created for each polyCone. A central radial field that pushes the Macaws up and away from the center of the scene is created and attached to each polyCone. And a turbulence field is created and attached to each polyCone, in case the motion needs to be more erratic (**figure 2**).

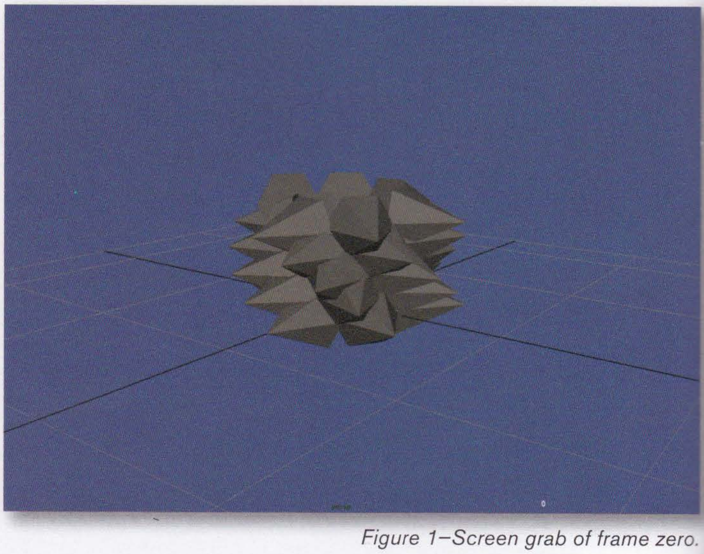


Figure 1–Screen grab of frame zero.

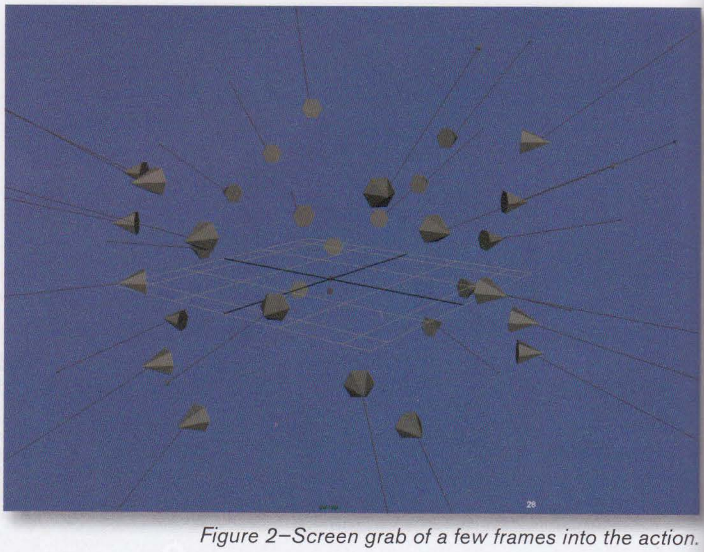


Figure 2–Screen grab of a few frames into the action.

The Splashflock script starts with a declaration of variables. The script requires an integer variable *$copy* for the number of copies of the polyCone that will be generated. A float variable *$distance* will designate the number of units between each copy.

```
int $copy;
$copy = 6;
float $distance;
$distance = 2.5;
```

Create a polyCone that will be the master node for each Macaw.

```
polyCone -name bird0 -r 1 -h 2 -sx 6 -sy 1 -sz 0 -ax 0 0 1
-tx 0 -ch 0;
```

The *"-name bird0"* flag also names the polyCone *"bird0."* Having the polyCone named in an easy-to-remember fashion makes script writing easier. Having the name sequence end in the number "0" is vital because this node is going to go through a few "for" loops. Programs tend to start sequences with "0," so I do the same. The *"–r"* denotes radius, the *"–h"* height. The *"-sx," "-sy,"* and *"–sz"* are subdivision settings. These are default. *"-ax 0 0 1"* orients the point of the polyCone along the Z-axis. This is because the expression that orients the cones does so along the Z-axis. *"-tx 0 "* and *"-ch 0 "* turn off texture and construction history, respectively, since these will not be used.

The polyCone that was just created is "selected" so it can be manipulated without having to use a *"-name"* flag on the following commands. So I did a duplicate and rotate 30 degrees on the Y-axis.

```
duplicate -rr;
rotate -r 0 30 0;
```

The *"-rr"* is the "Return Roots Only" flag. It is the default setting and must be included. The *"-r"* in the rotate command denotes relative rotation compared to the rotation of the previous object. Now the duplicate, which is named *"bird1,"* is selected. I run a "for" loop to make as many duplicates of the polyCone as is held in the *"$copy "*variable.

```
for ($i=1; $i<$copy; ++$i) duplicate -rr -st;
```

The duplicate command includes the *"-st"* flag, for "Smart Transform." This remembers the rotation of *"bird1"* and applies it to the following copies.

Next, the array of polyCones is distributed in a circle. The distance between each polyCone was set before as the float variable *"$distance."* A "for" loop cycles through the sequence of *"birds"* and moves each one based on a little math and its number in the sequence.

```
for ($i = 0; $i <= $copy; $i++)
{
    move (sin( $i) * $distance) (.1 * $i) (cos($i) * $distance)
("bird" + $i);
}
```

I use a variable for the distance between the Macaws since at this point I don't know how big the model will be in world space and I don't know how much space they'll need to take up on the screen in each shot. When I get to that point in the production, I'll be able to adjust accordingly without having to individually hand tweak an unknown number of Macaw models.

The birds will need a dynamic solver to do the calculations related to their individual position transformations. It's done with the following command:

```
rigidSolver -create -current -name birdSolver -
velocityVectorScale 0.5 -displayVelocity on;
```

The rigid solver is named "birdSolver" so I'll remember it, just in case I need to go back and alter its settings. *"-current"* sets *"birdSolver"* as the current solver. *"-displayVelocity on"* allows each rigid body to display a small arrow showing its direction, the length of that arrow indicating speed. "-velocityVectorScale" controls the size of that arrow.

By default, the solver binds to the position and rotation transforms of the active rigid body objects associated to it. For my Splashflock, I will want the birds to orient themselves in the direction they are flying. That will be handled by an expression, but for that expression to work, the solver will have to be disconnected from the rotation channels. The disconnect command by itself will not do this. But there is an option available in the solver to permit disconnection. And since I'm dealing with the solver at the moment, I might as well take care of this.

```
setAttr birdSolver.allowDisconnection 1;
```

Now to make each of the polyCones an active rigid body. This allows them to respond to fields. I'll use a "for" loop to cycle through the list of *"birds#,"* create a rigid body for each, and attach that rigid body back to the *"birdSolver"* rigid solver.

```
for ($i = 0; $i <= $copy; $i++)
```

This bracket begins the body of the loop. Then the rigid body is created.

```
{
    rigidBody -name ("rigid_bird" + $i) -active -mass 1 -
    bounciness 0 -damping 1.5 -impulse (sin($i * .1)) .1 0
    -solver birdSolver ("bird" + $i);
```

The *"-name"* flag names each rigid body *"rigid_bird"* and the number from the sequence. So "bird0" has "rigid_bird0," "bird1" has "rigid_bird1," and so on. The *"-impulse"* flag gives each bird a little kick at the beginning of the animation. For this I use *(sin($i * .1))* to start a spiral motion out from the center and a *.1* to push up. The *"-solver "* flag connects each *"rigid_bird#"* to the *"birdSolver."*

Continuing in the "for" loop, I disconnect the rotation channels of each *"bird#"* node from their associated "rigid_bird#".

```
    disconnectAttr ("rigid_bird" + $i + "rx.output")("bird" + $i +
    ".rotateX");
    disconnectAttr ("rigid_bird" + $i + "ry.output")("bird" + $i +
    ".rotateY");
    disconnectAttr ("rigid_bird" + $i + "rz.output")("bird" + $i +
    ".rotateZ");
```

Now each polyCone can rotate freely, but its position is dictated by its rigid body.

Since I'm still in the "for" loop, it's a good time to add a radial field to each polyCone to make them avoid one another.

```
radial -position 0 0 0 -name ("radial_bird" + $i) -magnitude .5
-attenuation 1
 -maxDistance 30.0;
```

The *"-name"* flag gives each a sequential *"radial_bird#"* name. The *"-attenuation"* flag causes the field's intensity to drop off based on distance. The *"-magnitude"* flag is set to .5 to cause the field intensity to drop off rapidly. *"-maxDistance"* makes the field cover only 30 units from the bird. Next, I point constrained the radial field to its associated bird.

```
    pointConstraint ("bird" + $i) ("radial_bird" + $i);
```

Now it's time to create an expression that will give each bird's path some randomness. Start with declaring a couple of variables that will create a random number on each frame.

```
    float $randX = rand(-0.1,0.1);
    float $randZ = rand(0.1,-0.1);
```

Using a variable called *"$expString,"* I write an expression for each *"bird#."* The first line is a comment.

```
    $expString = "//Set wander motion for each bird.\n";
```

The expression designates the current bird's *"impulseX"* attribute.

```
    $expString += ("bird" + $i + ".impulseX");
```

Equals sine times current time times the *"$randX"* variable, followed by a carriage return.

```
    $expString += " = sin(time * ";
    $expString += ($randX);
    $expString += ");\n";
```

The same math is applied to the *"impulseZ"* attribute.

```
    $expString += ("bird" + $i + ".impulseZ");
    $expString += " = (noise(time) * ";
    $expString += ($randZ);
    $expString += ");\n\n";
```

Continuing the expression, this part orients the bird to face in the direction of its motion. The first line is a comment.

```
$expString += "//Set orientation for each bird based on
(rigid_bird + $i).velocity array attribute.\n";
```

First create a float variable array.

```
$expString += “float $orient” + $i + “[];\n”;
```

Set the *“$orient#”* variable to the velocity attribute of its associated *“rigid_bird#”* node.

```
$expString += “$orient” + $i + “ = ` getAttr rigid_bird” + $i
+ “.velocity`;\n\n”;
```

Set the rotateX attribute to 0.

```
$expString += “bird” + $i + “.rotateX = 0;\n”;
```

Set the rotateY attribute to a specific degree using the “atan2d” based on the X and Z components of the *“rigid_bird#”*’s velocity attribute array.

```
$expString += “bird” + $i + “.rotateY = atan2d($orient” + $i +
“[0], $orient” + $i + “[2]);\n”;
```

And set the rotateZ attribute to 0.

```
$expString += “bird” + $i + “.rotateZ = 0;\n”;
```

Now create the expression using the *$exprString* variable.

```
expression -s $expString -name (“orientingExp” + $i) -
alwaysEvaluate true -unitConversion all;
```

The *“-name”* flag names the expression *“orientingExp#”* in association with its *“bird#”* node. And the close bracket finishes the “for” loop.

```
}
```

All this will generate an error message that isn’t indicative of any problem, but is a bit annoying. So let’s get rid of it.

```
cycleCheck -e off;
```

In the previous “for” loop, a radial field was created for each bird. Now that all the fields have been created, they need to be connected to the other rigid body objects. This is accomplished with a “for” loop within a “for” loop. The parent “for” loop establishes the number that will be used to denote the *“radial_bird#”* field that will be acted upon. The child “for” loop cycles through the whole list of *“rigid_bird#”* fields and initiates a connection to each.

```
for ($i = 0; $i < ($copy + 1); $i++)
{
    //Do connections
    for ($j = 0; $j < $copy; $j++)
    {
        connectDynamic -fields (“radial_bird” + $i)
(“rigid_bird” + $j);
    }
```

The current *“radial_bird#”* field does not need to be connected to its related *“rigid_bird#,”* so I did a little housecleaning and disconnected the two. The last bracket closes the parent “for” loop.

```
    connectDynamic -delete -fields (“radial_bird” + $i) (“rigid_
bird” + $i);
}
```

The final portion of this script creates a couple of global fields to push the Macaws up and away from the scene origin and give me the ability to add some more user definable randomness to the motion of the flock elements via a turbulence field. First I create a radial field just below the origin. From this coordinate, the force on each bird comes from slightly below its initial position, driving it upward and out.

```
radial -position 0 -1 0 -name “splash” -magnitude 0.4 -
attenuation 0;
```

Then I use a “for” loop to connect this field to all the *“bird#”* nodes.

```
for ($i =0; $i < ($copy + 1); $i++)
{
    connectDynamic -f splash (“bird” + $i);
}
```

And follow that by creating a turbulence field. The flags are all at default settings. I just want to have this handy in case the Creative Director would like me to make the flight paths of the Macaws a little more erratic.

```
turbulence -pos 0 0 0 -m 25 -att 0 -f 1 -phaseX 0 -phaseY 0
-phaseZ 0 -noiseLevel .05
        -noiseRatio 0.707 -mxd -1 -vsh none -vex 0 -vof 0 0 0
-vsw 360 -tsr 0.5 ;
```

Then a “for” loop to connect the turbulence field to each of the *“bird#”* nodes.

```
for ($i =0; $i < ($copy + 1); $i++)
{
    connectDynamic -f turbulenceField1 (“bird” + $i);
}
```

Now I have a script that produces a “Splash Flock” of stand-ins.

At this point, I turn my attention to the Macaw model. I was partially responsible for construction and fully responsible for texturing and rigging of the Macaw. For the purposes of this project, I opted for a flapping cycle controlled by an animation clip. I created a character set of all the joints in the skeleton. I created a four keyframe flapping cycle. Saved that as an animated clip. In the Trax Editor, I selected the clip and created a timewarp curve called ““*FlapCycle.*” Then I selected the *“FlapCycle”* curve and applied the *Pre-Infinity > Cycle* and *Post-Infinity >Cycle* commands to it. The wings now flap endlessly. I saved the model as *“Sbird_Flap_REF.ma.”* Incidentally, I use the Maya ASCII (.ma) format because I may need to edit the file in UltraEdit. This is purely a matter of convenience and personal preference. The “REF” in the file name is a bit of a note to myself that this file will be the source file referenced into my scene, rather than a straight import. The top node of the Macaw model is called *“Macaw_AllMacaw0,”* since it will be used in some “for” loops. The parent node of the skeleton is called *“Macaw_Master.”*

Starting with a new scene file, I ran the “SplashFlock.mel” script. I then referenced “Sbird_Flap_REF.ma” and duplicated it five times for a total of six Macaws. Now those each need to be centered on and attached to their associated *“bird#”* nodes. Time for another MEL script.

Making the “Attacher.mel” script was a much easier task. I went through the process by hand one time, copied all the MEL from the Script Editor to UltraEdit, and assembled it into a couple of “for” loops.

The first “for” loop cycles through the *“Macaw_AllMacaw#”* nodes. It applies a point constraint and an orient constraint to its *“Macaw_Master”* skeleton node connected to a specific *“bird#”* node. Start by declaring a variable for the number of Macaws in the scene.

```
int $copy;
$copy = 6;
```

Then the first “for” loop.

```
for ($i = 0; $i <= $copy; $i++)
{
```

Select the *“bird#”* node, followed by selecting the *“Macaw_AllMacaw#|Macaw_Master”* node that will be constrained to it.

```
select -r (“bird” + $i);
select -add (“Macaw_AllMacaw” + $i + “|Macaw_Master”) ;
```

Do a point constraint command and an orient constraint command.

```
pointConstraint -offset 0 0 0 -weight 1;
orientConstraint -offset 0 0 0 -weight 1;
```

The Macaw is facing 90 degrees left of the motion it will follow, so I set the offset attribute of the orient constraint. This could have been done in the previous command at the *“-offset”* flag. But here it is, instead.

```
setAttr (“Macaw_Master_orientConstraint” + ($i + 1) +
“.offsetY”) 90;
```

The Macaw’s posture does not quite work with the motion. It just needs to lean forward, into its flight path. So, I rotated it from its

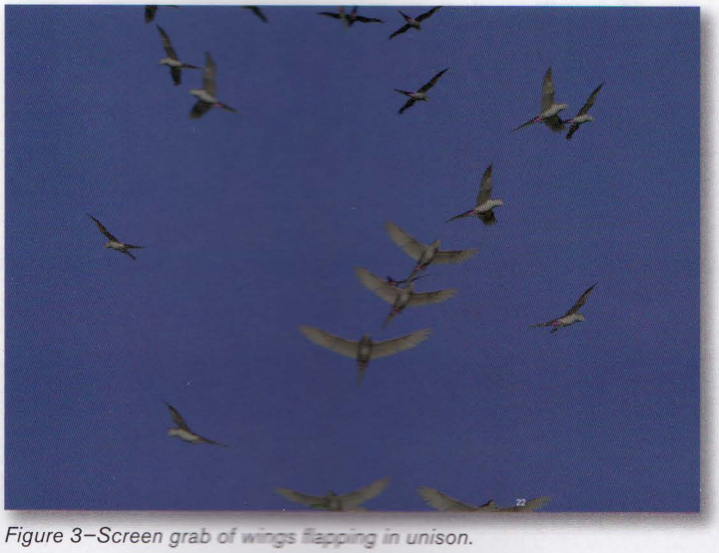


Figure 3–Screen grab of wings flapping in unison.

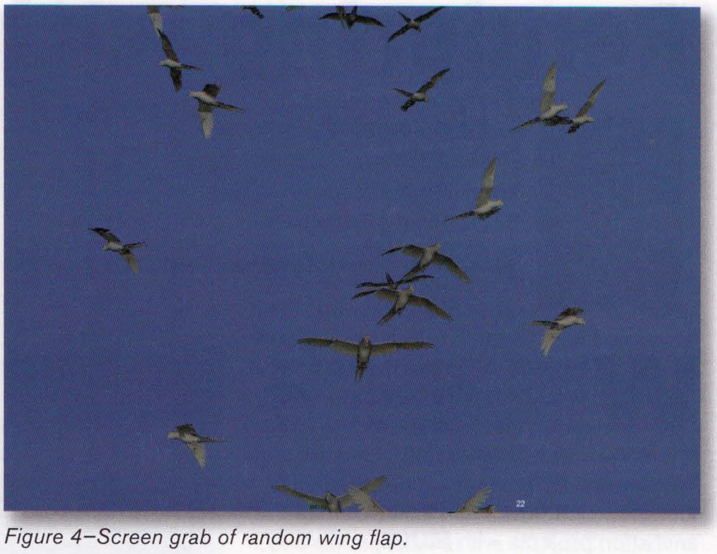


Figure 4–Screen grab of random wing flap.

pelvis minus 35 degrees on the local Z-axis. This is achieved with the *"-os"* flag.

```
select -r (“Macaw_AllMacaw” +$i + “|Macaw_Master|jPelvis”) ;
rotate -r -os 0 0 -35;
}
```

Now all the Macaws fly away, but their wings flap in unison (**figure 3**). Time to randomize that. Start by declaring a "random" float variable.

```
float $rand = rand (-15,0);
```

The *"rand"* command generates a random number. The argument *"(-15,0)"* tells *"rand"* to generate a number between –15 and 0. This sets the *"$rand"* variable to offset the flapping cycle of each Macaw somewhere between –15 and 0 frames. Now to cycle through the Macaws with a "for" loop.

```
for ($i = 0; $i <= $copy; $i++)
{
```

First select the current Macaw's *"FlapCycle_timeWarp"* curve. The curve starts at frame 0 and ends at frame 50. The *"-t"* flags in the *"selectKey"* command cause the whole curve to be selected.

```
selectKey -add -k -t 0 -t 50 (“FlapCycle_timeWarp” + $i) ;
```

Offset the curve with the *"keyframe"* command. The *"-timeChange"* argument applies the current value of *"$rand"* to that animation curve.

```
keyframe -animation keys -option over -relative -timeChange $rand ;
}
```

"$rand" is reset and the "for" loop goes on to the next *"FlapCycle_timeWarp"* curve in the series. Now the flock looks more natural and unsynchronized (**figure 4**).

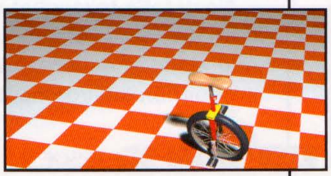
With a tested pipeline in place, I am now ready to produce splash flock elements for the shots. I start with a copy of the scenefile that has the final camera and final animated Dolphin. Render a playblast at output resolution and map that to an image plane in the camera window. Then, delete everything but the camera. Now I have a mock-up of the scene to work with. Follow that with the splash flock pipeline. I have to do it a few times, adjusting the *"$distance"* variable and the scale of the Macaws so they are completely obscured by the Dolphin at the first frame of the splash flock animation. Then select all the polyCones with *"select –r polyCone *;"* and hide them. Hide the image plane and do a playblast of the splash flock. Once I'm satisfied with the playblast, I e-mail a link to the Creative Director for approval. Upon approval, I put the scene into the render queue. The next morning, check the final animation, if it's OK, e-mail a link to the rendered files to the editor. I did that for two shots and it went smoothly.

I saved myself a lot of time, and my client a lot of money, by scripting instead of hand keyframing. The script worked reasonably well for the production and the Creative Director did not ask for tweaks, though I was prepared for that. I think the key to the successful implementation of MEL scripts in this production was keeping them rudimentary. The splash flock pipeline could have been completely scripted, with each of the MEL scripts I developed used as procedures. A simple user interface could have been integrated. But there were only two shots that required the splash flock, so keeping it simple was sufficient. And I got done in time to get to Mr. Chu's Pizza Tornado before they closed.

***Thatcher Bell** has been animating professionally for 14 years. His current favorite platform is a Toshiba M200 TabletPC.*

T-Splines: Better than NURBS

Introducing a new time-saving, revolutionary approach to modeling. T-Splines offers maximum freedom to the artist to create high-detailed models and is compatible with NURBS and subds. Order the Maya Plugin today at ***www.tsplines.com***—save $50 by using coupon: *hdri3dpromo*. Expires 6/30/06.



Rendering with LightWave 101:

Haulin' With MULE

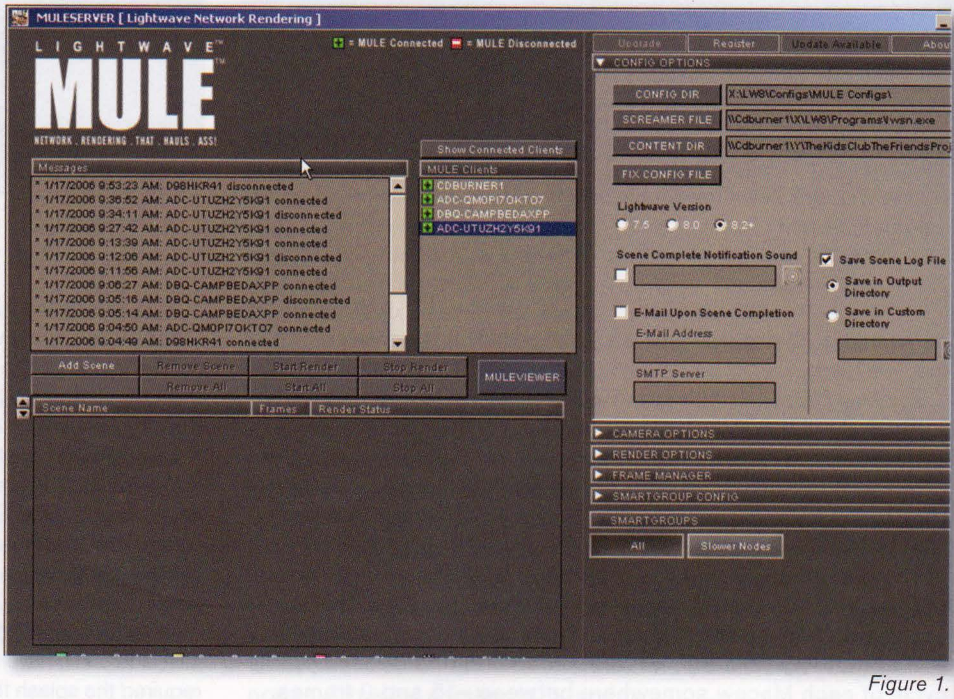
By Dale Campbell, Jr.
3D Animator
Dubuque, Iowa

When you are working on a project and are in full production, one of the many delays can be the rendering and test rendering of scenes. What I want to do is take you through the weekly process I go through on my animation projects with **MULE**, and show you how I saved up to 30% of production time using this program.

First, if you don't already own it, I encourage you to go out and get the previous issue of *HDRI 3D*. If there are questions about setting up your network for network rendering, please refer back to the previous issue

Also, if you have questions, please, feel free to contact me at *dale@positiveproductionsplus.com* or visit the epicsoft website at *http://www.epicsoft.net/products/mule/networkinstall.aspx*. They are good people and would be more than happy to answer any and all questions.

To get things rolling, let's fire up the MULESERVER (**figure 1**).



```
* 1/17/2006 9:53:23 AM: D98HKR41 disconnected
* 1/17/2006 9:36:52 AM: ADC-UTUZH2Y5K91 connected
* 1/17/2006 9:34:11 AM: ADC-UTUZH2Y5K91 disconnected
* 1/17/2006 9:27:42 AM: ADC-UTUZH2Y5K91 connected
* 1/17/2006 9:13:39 AM: ADC-UTUZH2Y5K91 connected
* 1/17/2006 9:12:06 AM: ADC-UTUZH2Y5K91 disconnected
* 1/17/2006 9:11:56 AM: ADC-UTUZH2Y5K91 connected
* 1/17/2006 9:06:27 AM: DBQ-CAMPBEDAXPP connected
* 1/17/2006 9:05:16 AM: DBQ-CAMPBEDAXPP disconnected
* 1/17/2006 9:05:14 AM: DBQ-CAMPBEDAXPP connected
* 1/17/2006 9:04:50 AM: ADC-QM0PI7OKTO7 connected
* 1/17/2006 9:04:49 AM: D98HKR41 connected
```

MULE Clients:
- 🟩 CDBURNER1
- 🟩 ADC-QM0PI7OKTO7
- 🟩 DBQ-CAMPBEDAXPP
- 🟩 ADC-UTUZH2Y5K91

CONFIG DIR: X:\LW8\Configs\MULE Configs\
SCREAMER FILE: \\Cdburner1\X\LW8\Programs\lwsn.exe
CONTENT DIR: \\Cdburner1\Y\TheKidsClubTheFriendsProj

Lightwave Version: ○ 7.5 ○ 8.0 ● 8.2+

Scene Complete Notification Sound ☐
☐ E-Mail Upon Scene Completion
E-Mail Address
SMTP Server

☑ Save Scene Log File
● Save in Output Directory
○ Save in Custom Directory

Figure 1.

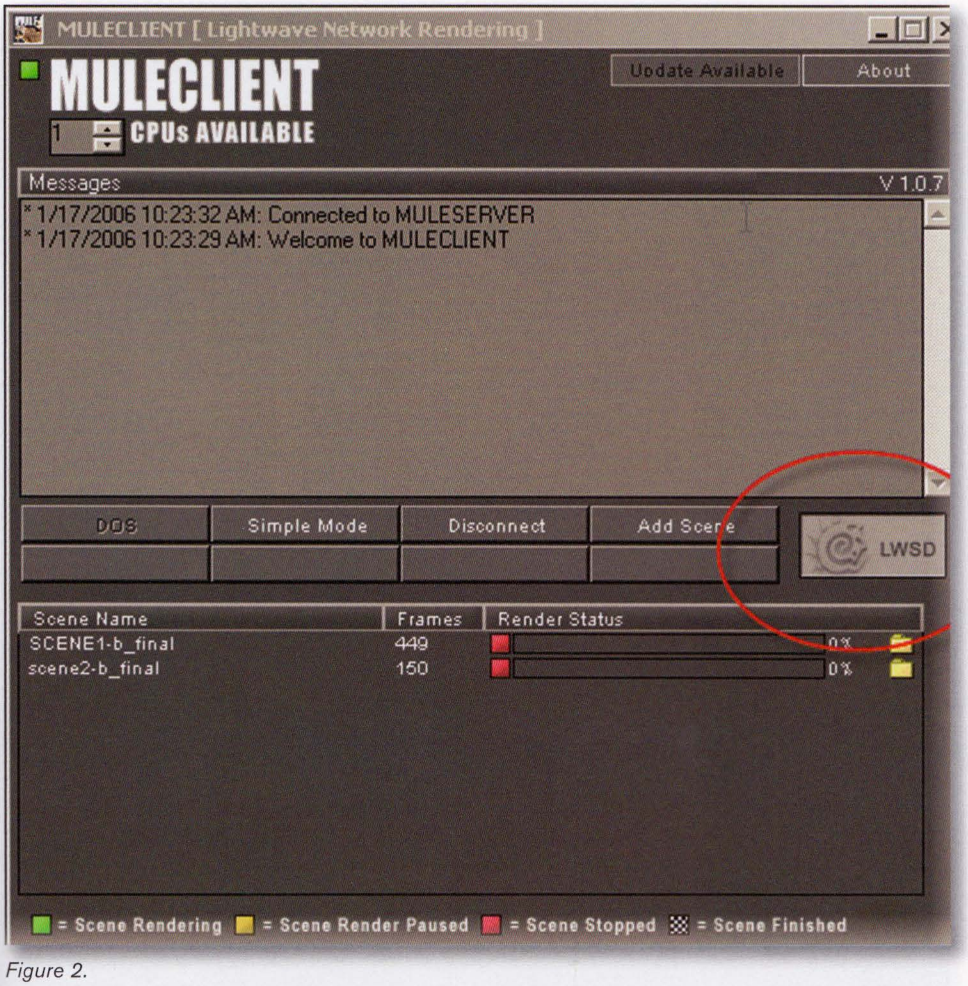
LightWavers will find the **MULE** controls to be familiar and simple; with render settings that users are already acquainted with. Color-coded visual cues, on-screen instructions, right-click menus, drag-and-drop functionality, an organized, logical layout, progress and status indicators, instant feedback, BAM! It's all right there in front of you and optimized for streamlining your workflow.

Remember from the previous article that each machine in your render farm will need to have the MULECLIENT program installed and running. Once again, the MULECLIENT needs no configuring. Only one instance of the MULESERVER is necessary for managing your entire render network. It is installed on a primary control system; this, in many cases, is the machine LightWave is installed on. Each additional system node in your network simply needs the **MULECLIENT**. Configuration and control of the whole network is managed from the main **MULESERVER** interface on the primary system.

Next, let's take what we learned from the last issue and apply it to a real world situation. In an animation production environment, problems and needs change, and unfortunately so does the scenario of the workflow. For me, last week there were issues with the deformation of the mocap data I am using in LightWave. This week I need quick renders to check the flow and timing of a scene and also frame renders to check things like the clothing of the character. Just a quick side note: The specs on my primary LightWave machine are an AMD 64bit processor 3200+ with 3gigs of ram, running for this tutorial WinXP Pro. This is a very powerful computer, but because of everything I mentioned above, the rendering time is still at a whopping 26 minutes a frame. Distributed rendering is the only answer. This is where MULE does the haulin'!

Let's break MULE down. A large part of what makes MULE so intuitive is that along with all its many features, it also mirrors all the rendering functions that LightWave 7.0 to 8.5 currently offers. The Camera Panel in **MULE** is a parallel of the Camera Panel within LightWave. **MULE** pulls all the information from the LightWave Camera Panel and gives you the ability to change parameters on the fly. **MULE** automatically holds your selected information within a proprietary file and renders your scene with those settings, while your original file remains unaltered. Using this powerful tool gives you the freedom to test Resolution Multipliers, Antialiasing Modes, or Reconstruction Filters without ever having to go back into the LightWave interface. **MULE**'s Camera Panel doesn't end there! Common in animation production, there will be multiple cameras, so select the camera you want to render from and send it off to render. Like I said previously, MULE hangs on to your selections, leaving your original intact.

What makes this feature extremely powerful is if, let's say you've created a pre-vis of an animation and the Animation Director has blocked out the same scene four different ways. Add the scene four times, change your cameras in each scene, hit "render all"–and then go back to working on your next

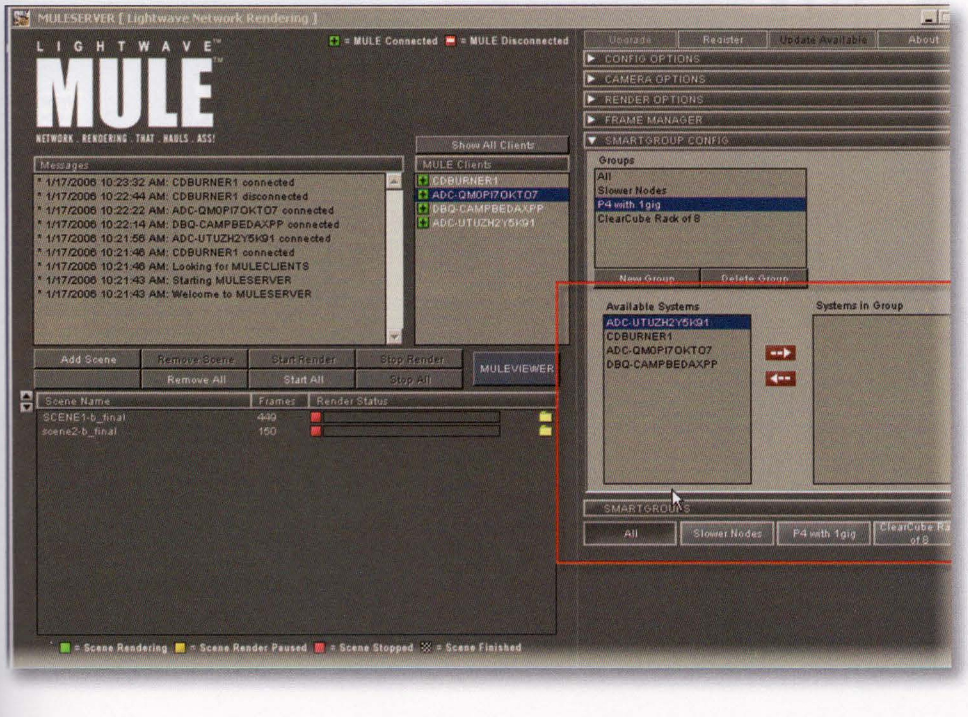

MULECLIENT [Lightwave Network Rendering]
MULECLIENT
1 CPUs AVAILABLE
Update Available
About
Messages
V 1.0.7
DOS
Simple Mode
Disconnect
Add Scene
LWSD
🟩 = Scene Rendering
🟨 = Scene Render Paused
🟥 = Scene Stopped
▨ = Scene Finished

```
* 1/17/2006 10:23:32 AM: Connected to MULESERVER
* 1/17/2006 10:23:29 AM: Welcome to MULECLIENT
```

Scene Name	Frames	Render Status
SCENE1-b_final	449	0%
scene2-b_final	150	0%

Figure 2.

Figure 3.


MULESERVER [Lightwave Network Rendering]
LIGHTWAVE™
MULE™
NETWORK . RENDERING . THAT . HAULS . ASS!
🟩 = MULE Connected
🟥 = MULE Disconnected
Show All Clients
Messages
MULE Clients
Add Scene
Remove Scene
Start Render
Stop Render
Remove All
Start All
Stop All
MULEVIEWER
Upgrade
Register
Update Available
About
CONFIG OPTIONS
CAMERA OPTIONS
RENDER OPTIONS
FRAME MANAGER
SMARTGROUP CONFIG
Groups
New Group
Delete Group
Available Systems
Systems in Group
SMARTGROUPS
All
Slower Nodes
P4 with 1gig
ClearCube Ra of 8
🟩 = Scene Rendering
🟨 = Scene Render Paused
🟥 = Scene Stopped
▨ = Scene Finished

```
* 1/17/2006 10:23:32 AM: CDBURNER1 connected
* 1/17/2006 10:22:44 AM: CDBURNER1 disconnected
* 1/17/2006 10:22:22 AM: ADC-QM0PI7OKTO7 connected
* 1/17/2006 10:22:14 AM: DBQ-CAMPBEDAXPP connected
* 1/17/2006 10:21:56 AM: ADC-UTUZH2Y5K91 connected
* 1/17/2006 10:21:46 AM: CDBURNER1 connected
* 1/17/2006 10:21:46 AM: Looking for MULECLIENTS
* 1/17/2006 10:21:43 AM: Starting MULESERVER
* 1/17/2006 10:21:43 AM: Welcome to MULESERVER
```

MULE Clients:
- CDBURNER1
- ADC-QM0PI7OKTO7
- DBQ-CAMPBEDAXPP
- ADC-UTUZH2Y5K91

Groups:
- All
- Slower Nodes
- P4 with 1gig
- ClearCube Rack of 8

Available Systems:
- ADC-UTUZH2Y5K91
- CDBURNER1
- ADC-QM0PI7OKTO7
- DBQ-CAMPBEDAXPP

Scene Name	Frames	Render Status
SCENE1-b_final	449	
scene2-b_final	150	

scene. This is a tremendous workflow advantage, and one that I utilize constantly in my pipeline.

Click around a bit in the MULESERVER interface, and you'll notice that the Render Options panel also mirrors LightWave's. This functionality gives you the opportunity to test out options such as Raytrace, Save RGB formats, Alpha Formats, First and Last frames, and Render Mode Options freely. Also under Render Options is an amazing feature aptly named FRAMESPLITTER PRO. I will go through this exceptionally powerful feature in greater detail later.

There's a nice little workflow feature in MULECLIENT called SCENEDROP™. Let's take a look at the MULECLIENT for a moment. There are currently three ways to load a scene in MULECLIENT: first is to click the Add Scene button, second is to right-click in the interface and select "load scene," and third—the quickest method, in my opinion—is by using the SCENEDROP function.

What I am going to do now is load "drag and drop" in a scene I am currently working on from my project "The Amazing Kids Club—Friends." Please load a scene of your own. In this scene, one of the characters has retrieved a loose ball and is dribbling downfield to pass the ball to a teammate who is open. He passes the ball and the teammate kicks and scores and the celebration begins. I have broken this down to seven small parts of the total scene. Also, to help in the speed of animating the scene, I have created block characters with little geometry to work the rig. This first render is to check the timing of the running, jumping, and kicking in the scene. Because they're quick changes, I leave my scene content window open and simply drag and drop my scenes into SCENEDROP (**figure 2**).

Not only does MULECLIENT benefit the user with feedback from the MULESERVER program installed on their system, but also, users on any of the client machines can add additional projects to the SCENE-Q™ for rendering with SCENEDROP. This is definitely a "big studio" feature.

Let's move on to another feature called SMARTGROUPS™. SMARTGROUPS allows you to categorize your available render clients into groupings that are best suited for specific types of render tasks (**figure 3**). More powerful systems can

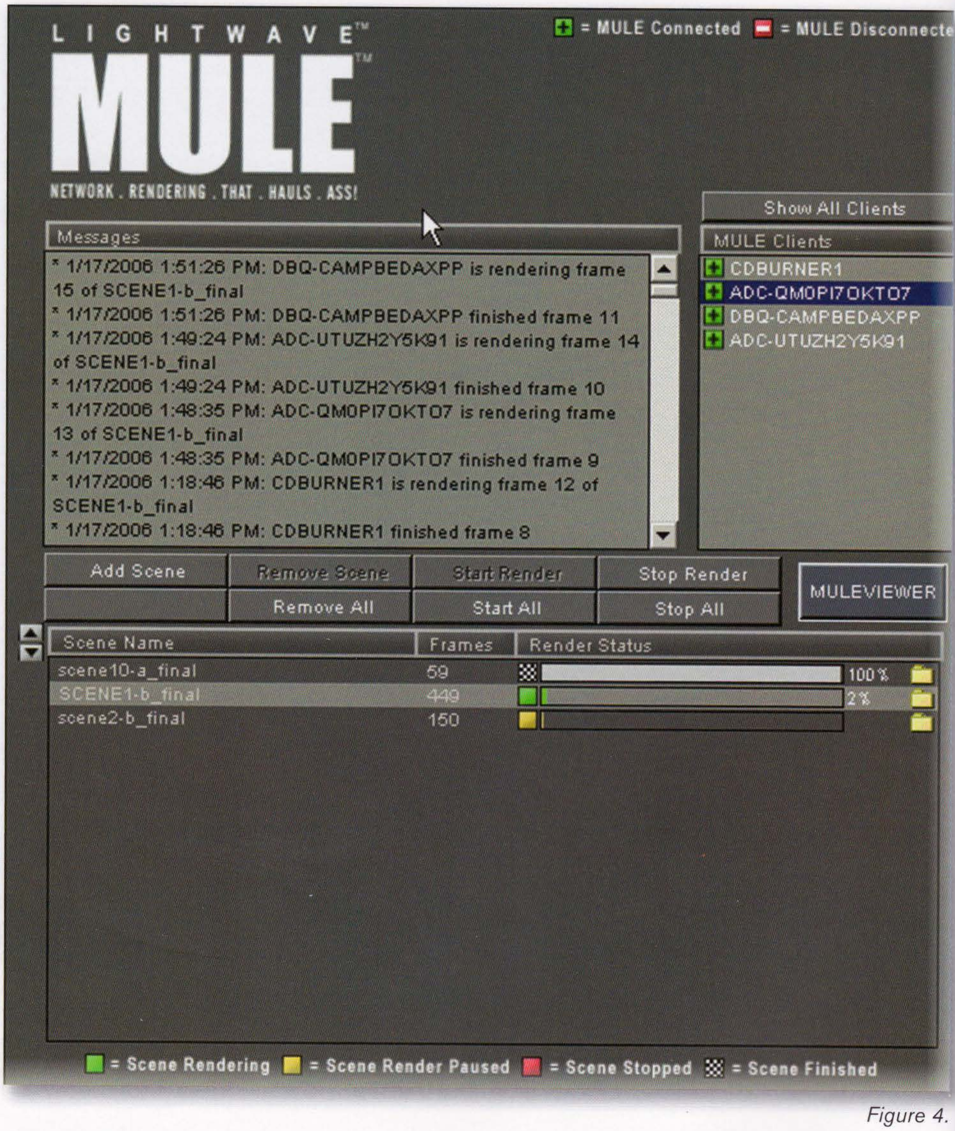
be designated for higher-resolution images or scenes that require intensive calculation, while your moderately equipped machines can be reserved for less taxing work that won't choke the overall render process. This powerful feature of MULE can have a great impact on your render time and quality.

Select the SMARTGROUPS CONFIG tab, click on the "New Group button," and type in a name for your group. You should see all your nodes, " MULECLIENTS," listed below in the available systems window. Select the systems you want for that group, click the red arrow button, and they will be transported to the "Systems In Group" window. They now are connected to the button you see on the bottom of the MULE interface. I like to set up my groups based on the power of my machines. I have talked to other animation companies that utilize MULE for their distribution rendering, and not only do they use their animation systems, but also any available system in the entire office. To maintain organization, they will name their systems based on whom they're stealing the rendering power from.

We are now ready to move to the SCENE-Q™ area. The SCENE-Q render manager is unprecedented in its responsive capabilities for efficiently coordinating your network render resources and the workload to be processed.

MULE and its SCENE-Q (**figure 4**) puts the controls at your fingertips. There are no obstacles to inhibit you from adding multiple scenes from multiple directories to be rendered, to cancel renders on the fly, for re-prioritizing the render cue, for adding new network clients on the fly, removing clients, and sending or re-routing specific tasks to specific systems. Whatever actions you may want to take to get the best renders, MULE is at your command. You can also just load your scene and let MULE do the work.

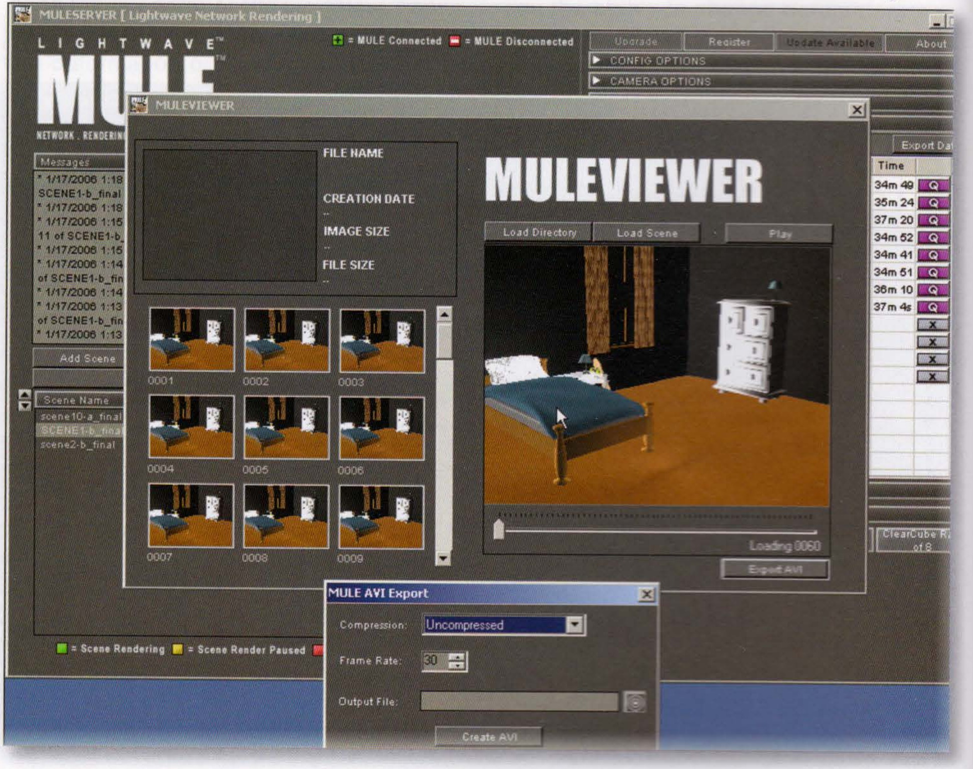
I really love the fact that while I continue to work and tweak my scenes, MULE will continue to render my scenes and at any given time I can see a preview of them. In fact, let's check out the MULEVIEWER™ now. With the MULEVIEWER you can look at frames as they are rendered–without having to wait for your whole project to complete and without the system overhead of launching an external image program. We can playback our animated scenes at any time for advance preview (**figure 5**). You can even play or scrub through the completed portion of an animated scene while the rest of the scene is still being rendered!


LIGHTWAVE™
MULE™
NETWORK . RENDERING . THAT . HAULS . ASS!
🟩 = MULE Connected
🟥 = MULE Disconnecte
Show All Clients
MULE Clients
CDBURNER1
ADC-QMOPI7OKTO7
DBQ-CAMPBEDAXPP
ADC-UTUZH2Y5K91
Messages
Add Scene
Remove Scene
Start Render
Stop Render
Remove All
Start All
Stop All
MULEVIEWER
🟩 = Scene Rendering
🟨 = Scene Render Paused
🟥 = Scene Stopped
▨ = Scene Finished

```
* 1/17/2006 1:51:26 PM: DBQ-CAMPBEDAXPP is rendering frame
15 of SCENE1-b_final
* 1/17/2006 1:51:26 PM: DBQ-CAMPBEDAXPP finished frame 11
* 1/17/2006 1:49:24 PM: ADC-UTUZH2Y5K91 is rendering frame 14
of SCENE1-b_final
* 1/17/2006 1:49:24 PM: ADC-UTUZH2Y5K91 finished frame 10
* 1/17/2006 1:48:35 PM: ADC-QM0PI7OKTO7 is rendering frame
13 of SCENE1-b_final
* 1/17/2006 1:48:35 PM: ADC-QM0PI7OKTO7 finished frame 9
* 1/17/2006 1:18:46 PM: CDBURNER1 is rendering frame 12 of
SCENE1-b_final
* 1/17/2006 1:18:46 PM: CDBURNER1 finished frame 8
```

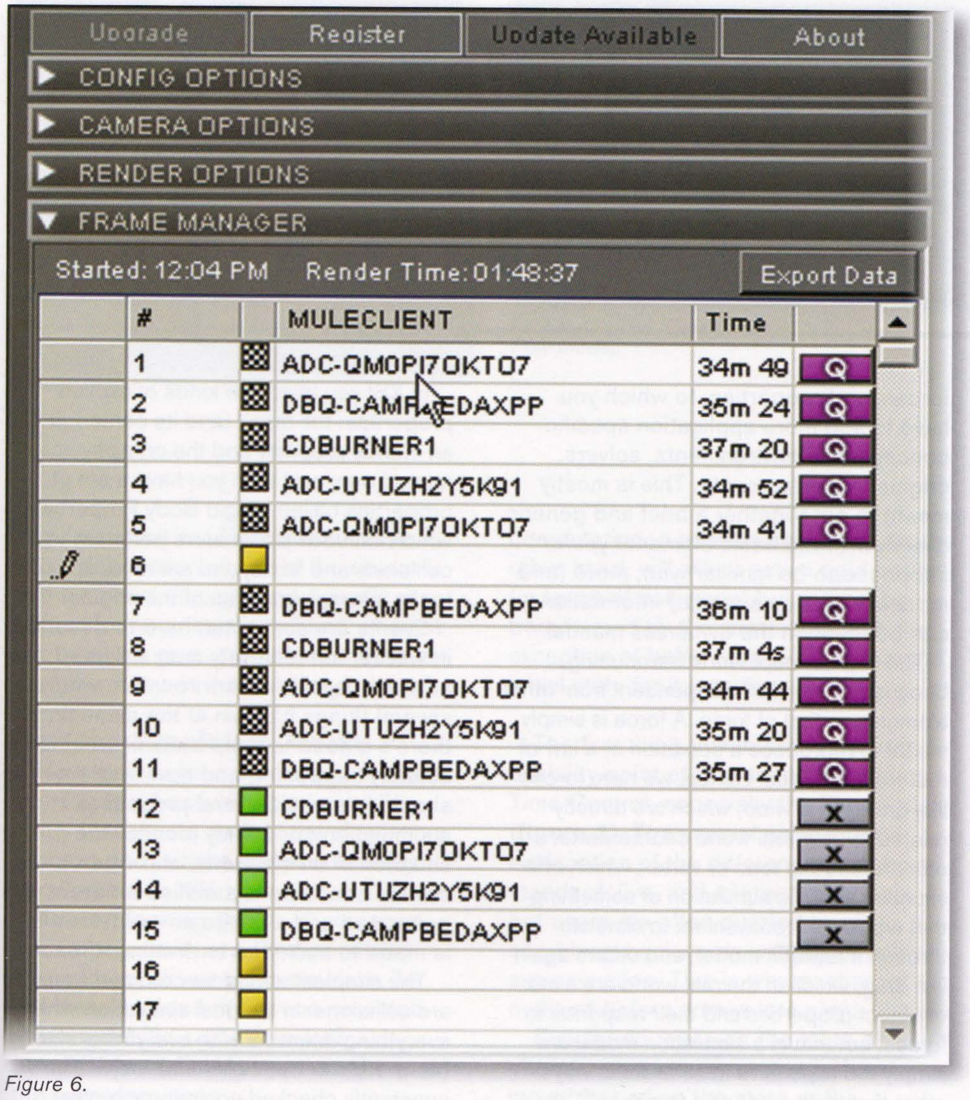
Scene Name	Frames	Render Status
scene10-a_final	59	▨ 100 %
SCENE1-b_final	449	🟩 2 %
scene2-b_final	150	🟨

Figure 4.


MULESERVER [Lightwave Network Rendering]
LIGHTWAVE™
MULE
🟩 = MULE Connected
🟥 = MULE Disconnected
Upgrade
Register
Update Available
About
CONFIG OPTIONS
CAMERA OPTIONS
MULEVIEWER
FILE NAME
CREATION DATE
IMAGE SIZE
FILE SIZE
MULEVIEWER
Load Directory
Load Scene
Play
0001
0002
0003
0004
0005
0006
0007
0008
0009
Loading 0060
Export AVI
Export Da
Time
34m 49
35m 24
37m 20
34m 52
34m 41
34m 51
36m 10
37m 4s
ClearCube R
of 8
Messages
Add Scene
Scene Name
scene10-a_final
SCENE1-b_final
scene2-b_final
🟩 = Scene Rendering
🟨 = Scene Render Paused
MULE AVI Export
Create AVI

Compression: Uncompressed
Frame Rate: 30
Output File:

Figure 5.



Started: 12:04 PM Render Time: 01:48:37

	#		MULECLIENT	Time	
	1	▨	ADC-QM0PI7OKTO7	34m 49	Q
	2	▨	DBQ-CAMPBEDAXPP	35m 24	Q
	3	▨	CDBURNER1	37m 20	Q
	4	▨	ADC-UTUZH2Y5K91	34m 52	Q
	5	▨	ADC-QM0PI7OKTO7	34m 41	Q
✎	6	🟨			
	7	▨	DBQ-CAMPBEDAXPP	36m 10	Q
	8	▨	CDBURNER1	37m 4s	Q
	9	▨	ADC-QM0PI7OKTO7	34m 44	Q
	10	▨	ADC-UTUZH2Y5K91	35m 20	Q
	11	▨	DBQ-CAMPBEDAXPP	35m 27	Q
	12	🟩	CDBURNER1		X
	13	🟩	ADC-QM0PI7OKTO7		X
	14	🟩	ADC-UTUZH2Y5K91		X
	15	🟩	DBQ-CAMPBEDAXPP		X
	16	🟨			
	17	🟨			

Figure 6.

It's also possible to export your image sequences as .avi files. Simply select the Export AVI button at the bottom of the MULEVIEWER interface, and you will be presented with a number of compression settings. Choose your desired compression format, frame rate, and location, and MULEVIEWER will compose an .avi file for you on the fly.

As I write this tutorial I am rendering the animation I spoke about above. The animation timing is to my liking, and now I will move on to the hi-res models for final animation, texturing, and lighting. I will be using radiosity along with cloth simulation, hypervoxels for dust, etc. Rendering out a single frame takes 26 min on a single machine. This is where I will tap into the power of FRAMESPLITTER PRO.

I have 45 MULECLIENTS on my system. This means I can take my render of 26 minutes and divide that by 45. With some machines being a bit slower, and some faster, I believe the render times even out. So now I'll select my desired frame, select the Output directory, select the amount of slices and my file type. My render time has just been cut down to approximately 57 seconds per slice, popping up in the preview window in about a minute and a half. I will also be using this feature when creating all my print material. At 300 dpi some of these images will get pretty huge, but FRAMESPLITTER PRO makes them manageable.

Now there are times when I notice early on that I made a mistake or there was a glitch in my scene (e.g., poor camera position, lighting problems, etc.). It's in those times that I have to use Frame Re-Q. The reliable FRAME Re-Q functionality enables you to pinpoint and re-send just your corrupted frames for rendering (**figure 6**). No need to unnecessarily re-render the already pristine elements of your project!

There are so many other features and elements that I love in MULE, like the ability to add LightWave scene files from different directories. Also, changing scene priority on the fly is a killer feature, especially when you consider that SMARTGROUPS follow along with the scene and as soon as a scene is completed it changes to that group without error.

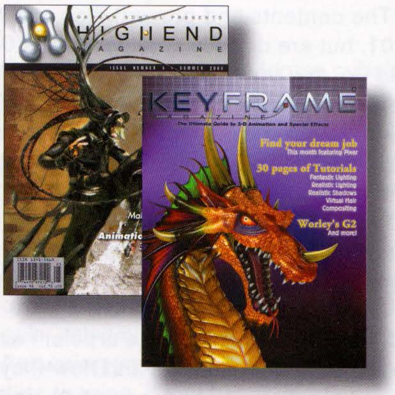
That's very cool and very important. Well, my friends, those are the basics of MULE 1.0.

In closing, I recently had a chance to review MULE 2.0, and there are a couple of highlights I want to point out. The new MULE 2.0 interface is beautifully designed and I get the feeling that a lot of thought and time was put into the workflow and user interactivity. MULE 2.0 now works under 32- and 64-bit systems. There are also unlimited groups and the ability to add unlimited processors on a single MULECLIENT.

Other new features include Dual Core Support, Batch Render options, Send from LightWave functions, Timed Scheduling, Scene History with recall, View Loaded Plugins, and Window Expansion.

The interface window panes can be resized and expanded to take over the entire interface. Also, you'll find massive upgrades to the MULEVIEWER and MULECLIENT. These are just some of the exciting features. You can check them out for yourselves soon. Word from epicsoft is it'll be released at the beginning of the quarter.

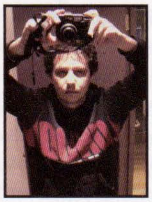
***Dale Campbell** lives in Dubuque, Iowa, with his wife Karin. They have 3 children: Tiphanie, Felecia, and Dale III. They have 2 grandsons, Isaiah and Josiah. He says, "These are my most beautiful renders." God Bless!*



ONLINE SPECIAL

Inventory Clearance Sale
www.dmgpublishing.com

Rigid Bodies and Spring Networks 101



By Raffaele Fragapane
Lead Character TD
Adelaide, Australia

When working in production on character assets, not a week goes by without some form of secondary motion or dynamics popping up in a meeting or in a requests list. While a great amount of literature has been published about how to trick your way around very specific problems, I've found there's very little information out there describing the fundamentals behind many dynamic systems and what can be achieved with them if you can think out of the box a bit.

In this article, rather then providing a single setup that will–maybe–get you out of the mud once in your professional life, I'll try to bridge the gap between technical knowledge and implementation, and hopefully give you some food for thought for your next setup, or at least shed some light on notions that are often overlooked by novice technical directors themselves.

The contents and presentation use XSI 5.01, but are directly applicable to 4.0 and up, and the notions themselves are applicable to pretty much any application that has basic dynamics functionality and a set of dynamic constraints like springs and hinges.

I also suggest that you keep the manuals open and at hand, as a quick search or read of some of the contents in there will often complement the information provided in the article. I aim to illustrate some basics and how they apply to simple secondary motion rigs, not to replace the actually quite good information provided by the manuals.

The State of Affairs (the Theory)

Staples in most simulation systems (we'll mostly deal with Rigid Body Dynamics from now on, shortened to RBD) are the concepts of state, events, forces, and properties, to which you have to add more application specific concepts like environments, solvers, engines, operators, etc. This is mostly meant to put together a brief and generic overview of the basic elements you're supposed to be familiar with; more (and more application specific) information can be found in the dynamics manual.

The first concept worth introducing, because it is almost independent from other elements, is that of force. A force is simply meant to reproduce a condition or a law of real world physics. Hence, we have forces like gravity and wind, which are directly related to their real world counterparts; other forces like toric or vortex, which offer situation specific simulation of something that would be inconvenient to simulate through a realistic model; and others again like drag, which in the real world are a mix of object properties and their response to forces, but that in a simulation model are simply too expensive to simulate in ways other than with a specific force.

NOTE: Forces (which you can get from the *Simulate > Get > Force* menu in XSI) are always calculated in global space; placing them in a hierarchy and animating them is possible, and often needed, but be aware that they will always be calculated at the end of the transformations chain, like if they were in global space. So, a gravity force child of a null rotated around the X-axis by 45 degrees could very well have its local SRT zeroed out, but it will still affect the environment as if it were rotated itself.

The properties of an object, in the generic sense, are parameters that will determine how that object interacts with the environment, meaning how it will affect, and be affected by, forces and other objects. In real world physics you have several properties that describe an object, but on the simulation side of things only one is emulated, which is its mass. It's a highly simplified simulation model, as the mass and its center is little more than a multiplier in the equations of the forces and collisions, but it's a necessary evil to keep the setup simple and the computation affordable.

In XSI you have two kinds of explicit properties: the mass (and its center) is an inertial property and the only physics related one; and then you have a set of properties called "Rigid Body Properties," which includes parameters pertinent to collisions and limits, and are more specific to the internal workings of the engine.

Events are somewhat hard to describe in words, but relatively easy to grasp as a concept. In an environment where several things happen at the same time, there's a need to keep track of what is actually happening and how elements should interact. Several processes in a simulation constantly monitor the situation of all elements, and when a certain condition is satisfied an event is generated and a call to an event handler is made to make the environment react.

The simplest and most common example are collisions: in an ideal simulation where everything interacts with everything else (no groups or optimizations), objects are constantly checked against each other for intersection. When such check does return a positive result (objects colliding), a collision event is triggered, the appropriate handler is called, and eventual additional calculations are made to affect the trajectory of the objects and so forth.

NOTE: All of the above–forces, events, and inertial properties–contribute to generate other properties for the object, which are acceleration and velocity along a motion vector; such properties can't be altered (as finding them is the objective of the simulation in the first place), but can be limited through four optional parameters in the RBD properties of an object.

All of the above contributes to generate the basic building block of a simulation, and the last (and arguably most important) fundamental we want to pin down, a state. Every object in a simulation, at any given time, is ultimately defined by its state; and a state is nothing but a set of information, including the object's transformations, its motion vector, and its linear and

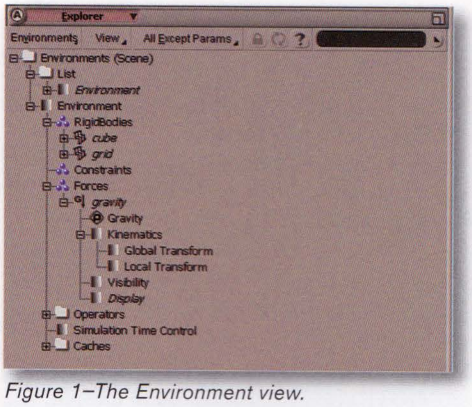

Explorer
Environments
View
All Except Params
?
Environments (Scene)
List
Environment
Environment
RigidBodies
cube
grid
Constraints
Forces
gravity
Gravity
Kinematics
Global Transform
Local Transform
Visibility
Display
Operators
Simulation Time Control
Caches

Figure 1–The Environment view.

angular velocities. All of this sums up into a description of where the object is at the current discrete unit (usually at a certain frame or subframe) and where it will be at the next unit before checks are performed to see if events take place.

How the Simulation Works

Now that we have introduced the base building blocks of a simulation, we're ready to look at a draft of the simulation process and to have a look at how it's handled inside XSI.

To start looking into it, all you need is a Cube primitive, a Grid primitive, and a force, all set up to simulate. To get that, just load the scene provided at *www.hdri3d.com/resources* (01_RBDbasicEnvironment.scn) or run the following steps:

1. Get a poly cube of size one and position it somewhere above the origin by a few units.

2. Make the cube an active rigid body with *Create > Rigid Body > Active Rigid Body* in the simulate panel.

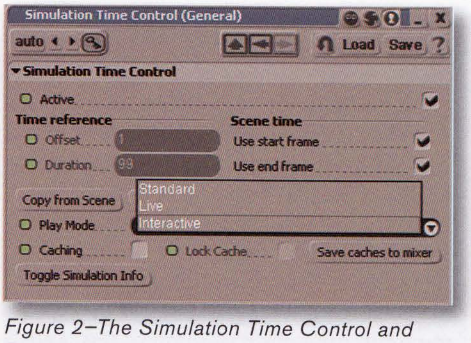
3. Get a default poly grid primitive.

4. Make it a passive rigid body with *Create > Rigid Body > Passive Rigid Body* in the simulate panel.

5. Get a gravity force with *Get > Force > Gravity* in the simulate panel.

This is enough to have an environment and properties to have a look at. If you set an explorer view's scope to Environment, you will see something similar to **Figure 1**.

NOTE: An active rigid body is a rigid body that is affected by forces and other bodies, and during the simulation has its transformations overridden by the computation; a passive rigid body, on the other hand, will affect active rigid bodies (in terms of collisions) but will not be transformed by the simulation (with the exception of being able to use a forceful initial state for its starting pose).


Simulation Time Control (General)
auto
Load
Save
?
Simulation Time Control
Copy from Scene
Standard
Live
Interactive
Save caches to mixer
Toggle Simulation Info

○ Active ✓

Time reference

○ Offset 1

○ Duration 98

Scene time

Use start frame ✓

Use end frame ✓

○ Play Mode Interactive

○ Caching ☐ ○ Lock Cache ☐

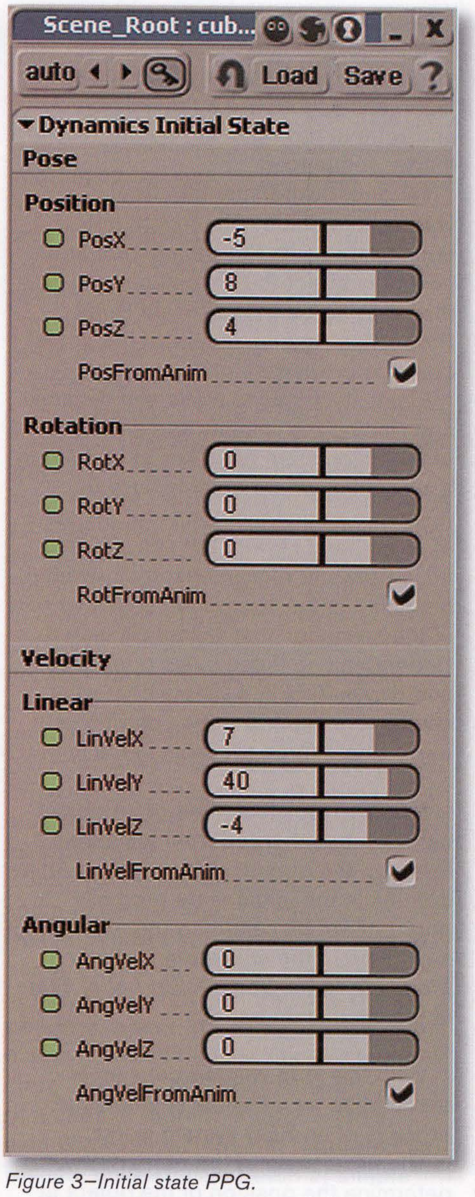
Figure 2–The Simulation Time Control and Play mode.

The first thing that interests us from a usability point of view is the Simulation Time Control, especially the Play mode (**figure 2**). The manual gives a good explanation of the differences between standard, live, and interactive mode, but users are often puzzled by why scrubbing backwards will seldom play any animation. This is where what we outlined before comes into play.

Simulation relies on the state of objects, and will compute the "next move" based on the state at the previous frame or subframe; hence, why a simulation that isn't cached can only be played forward. Setting the simulation to live and playing in loop, if the simulation isn't too heavy (which hardly ever is the case when RBDs are used for rigging), will allow play with parameters and the transformations of objects while the simulation is running, so interactive is what we'll keep our play mode in for the time being.

Playing our current simple scene will result in the cube falling straight on the grid, bouncing a couple times, and then sitting there just being uninteresting. Here's where the state concept can be extended to that of initial state.

Every object involved in a simulation is supposed to have a state at the beginning of the simulation, and every subsequent state is computed from that starting point. Very often people use a pre-roll, either hand animated or simulated, and then start rendering/capturing the animation only from a frame they are happy with, but that is very often unnecessary. Every rigid body, passive or active, has a "Dynamics Initial State" property set that can be tweaked at leisure.


Scene_Root : cub...
auto
Load
Save
?
Dynamics Initial State

Pose

Position

○ PosX -5

○ PosY 8

○ PosZ 4

PosFromAnim ✓

Rotation

○ RotX 0

○ RotY 0

○ RotZ 0

RotFromAnim ✓

Velocity

Linear

○ LinVelX 7

○ LinVelY 40

○ LinVelZ -4

LinVelFromAnim ✓

Angular

○ AngVelX 0

○ AngVelY 0

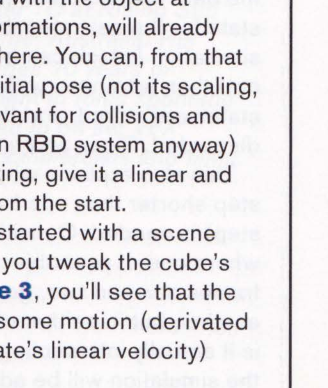
○ AngVelZ 0

AngVelFromAnim ✓

Figure 3–Initial state PPG.

Open the property page for the cube's initial state and you'll notice that its pose, unless you started with the object at zeroed out transformations, will already have an offset in there. You can, from that panel, tweak its initial pose (not its scaling, as that is only relevant for collisions and isn't affected by an RBD system anyway) and, more interesting, give it a linear and angular velocity from the start.

Assuming you started with a scene similar to mine, if you tweak the cube's Values like **Figure 3**, you'll see that the cube starts with some motion (derivated from the initial state's linear velocity)



and bounces off the grid. If now you also add some angular velocity, the state will also make the cube start spinning in its flight, much more like a tossed object, and without having to set up pre-rolls or longer sequences.

I suggest experimenting freely with simple setups like this to get comfortable with the parameters and how they affect the simulation, and then moving on to the "Rigid Body Properties" to also play with the actual collision properties of the objects.

Elasticity, frictions static and dynamic, and limiting the velocity and acceleration of the objects are often key to stabilizing a simulation where objects have to respect certain trajectories for the sake of using hand keying, but still simulate believably and remain onscreen without bouncing in the stratosphere for that part of the animation that will be simulated. Understanding these basic principles and parameters in a simple context is much more important than playing with every single force available in the menu.

NOTE: The initial state of an object can also be inherited from animation living on the object(s) (the ---FromAnim options in the initial state property page) or set by selecting the object(s)at any frame in the animation and using *Modify > Rigid Body > Set Initial State* in the simulation panel.

With this long-winded but simple explanation we can now see how the process of an RBD system works.

The initial state of the object(s) will determine the position of the object at the next step (frame or subframe). Before the simulation is advanced, the system checks for events in that new state. If one (like an intersection with an object or a force field) returns a positive check, the motion vector and the event ID are sent to a handler for that type of event(s), the data is computed, and then a new state for the next step is generated. The simulation (which can be cached or not) then advances with this new state, states are stored, and the previous ones discarded. Rinse and repeat.

This also explains why sometimes a step shorter than one frame (subframe step) is needed. Consider a situation where one of your objects, between frames one and two, goes through another, but in neither of the two states is it actually intersecting. In such a case, the simulation will be advanced as if nothing happened, which (especially in densely populated environments) might not be what you want. In such cases, using steps with a grain finer than a frame is needed.

Last but not least, don't try to commit to memory all of the above as if it were gospel. The truth of such systems is that there are many tricks and optimizations happening under the hood, and some of the event handlers or event checks are not as simplistic as the ones in the example; rather, consider it a simple mapping of the concept of state before/state after systems. It was, after all, meant to be an introduction to the real meat of this article...

Rigid Body Constraints and Why You Should Care

Now that we have established a few base concepts, and that hopefully you've played a bit with the excellent new RBD system in V5, we can talk about that set of special constraints that comes with such systems.

For the record, while ODE (the RBD system in V4.x) wasn't nearly as fast or precise as PhysX in V5, the RBD constraints were fairly solid and usable in ODE. So, all that follows easily applies to older versions as well.

As the definition of RBD is manipulating, according to certain rules, bodies that don't deform but are only transformed in pose, part of these toolsets usually is a slew of constraints to correlate two or more such objects by a link of some kind. The constraints available in both PhysX and ODE are hinge, spring, ball&socket, slider, and fixed. They can, to some extent, all be useful for rigging purposes, but what we're most interested in now is the spring constraint.

The spring constraint pretty much does what it says on the label—it connects two objects as if they were connected by a spring, and if one is fixed (a passive rigid body) and the other is free to move, you can already get some interesting effects out of it, but you'll be facing the problem of hard to predict rotations and motion going into a hyper-kinetic state and shooting out into the void.

Before we move on, let's set up a very simple scene to get accustomed to a couple tools we'll use for the next step. Load Scene 03_RBDsingleSpring.scn or perform the following simple steps:

Get two cubes—we'll call them cube1 and cube2.

Make cube1 a passive rigid body (to be involved in a simulation, even if being unaffected by it, an object needs to be a rigid body) and cube2 an active rigid body.

With nothing selected use *Create > Rigid Body > Multi Constraint Tool > Spring* and first left-click on cube2, then middle-click on cube1, and last right-click somewhere in the viewport to exit the tool.

Set the spring length to the distance between the two objects.

Set your simulation's play mode to Interactive.

Play with the scene moving around cube1 and feel free to tweak the spring parameters or make the simulation standard or live and play with parameters while playing it in loop.

NOTE: I would have liked to go into details about actual spring solvers and the parameters presented in XSI—parameters like strength and dampening—but the word count is my enemy and I'll have to defer that to the manual.

Now there probably are a couple of things you'll have noticed. First, if you're in interactive play mode, as soon as you stop transforming cube1 all activity suddenly stops. This is normal, as the interactive mode will do its best to follow what you're doing and make the RBD behave consequently; however, as this play mode monitors your actions and not a proper sequence of states to determine velocities, as soon as you stop interacting, the objects will be considered to have no state to be passed on.

Also, because the system is interactive, the stepping of the monitoring process can't be guaranteed. If your CPU is under severe stress while you're moving the object, two state steps could be read at one second from each other, making your mouse movement read as super fast, or as nearly irrelevant if you bounced back and forth, and generating odd velocities. The interactive mode is great for playing around with the initial steps of your setup, but don't rely on it for anything other than playing if you can avoid it.

The second thing you'll have noticed is that, if you tweaked the parameters in the "wrong" range and maybe moved things around a bit too much, your object might have disappeared into outer space, to be returned to its initial state only after flicking a frame forth and then one back. This is because a single spring can only do that much for you.

Last but not least, why the multi constraint tool and not the simple constraint one? This will become clearer with the next part, and with the amount of constraints we're going to create, but even if you were to create a single constraint, I'd still advocate the multi constraint tool. It works in a very simple manner, allowing you to pick in sequence several end points and start new springs without re-starting the tool.

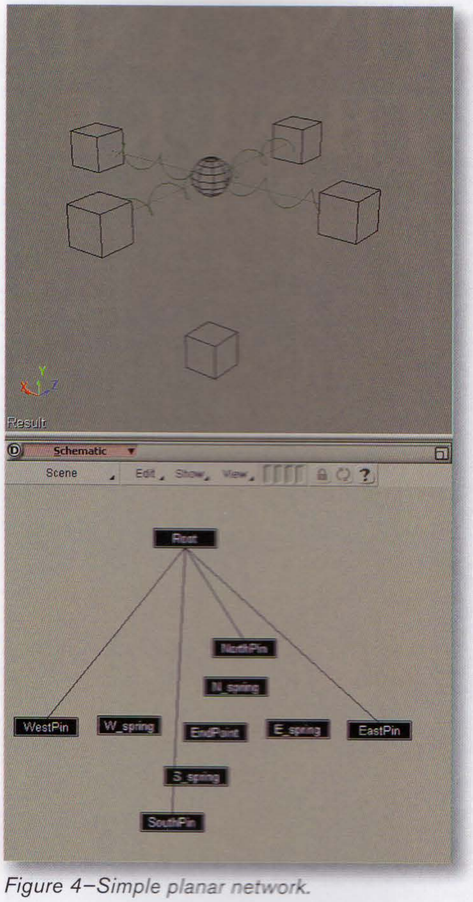


Figure 4–Simple planar network.

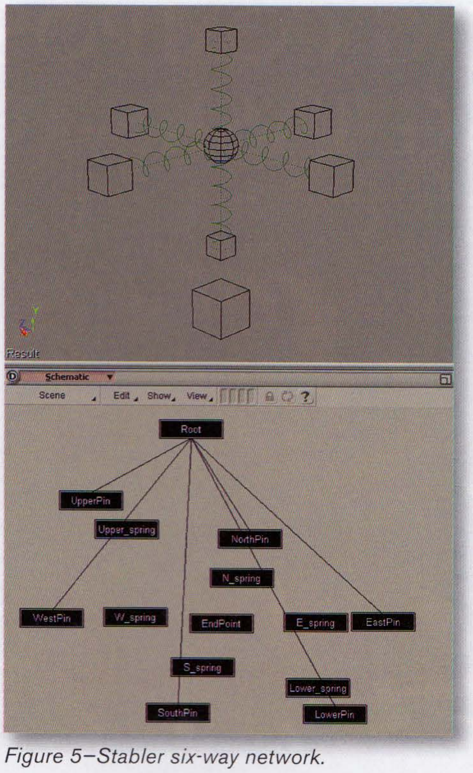


Figure 5–Stabler six-way network.

Some help on what the buttons do, as for everything else in XSI, is always presented at the right of the bottommost bar in the interface. But in case you are an animator and constantly smear your monitor with markers, you can first select with a left-click the object that will be driven by the spring, then middle-click on the object you want to attach to, and last, when you've drawn all the springs you need, right-click in the interface to exit the tool.

Let's move on.

Enter Spring Networks

By now you've probably heard the words "spring network" at some point, but not everybody is aware of how simple and useful they can be, or how easily implemented.

While a single spring, even with the best dampening, can easily turn unpredictable, using several opposing springs that balance each other will often result in more stable and predictable results. This is, after all, how most of the soft body and cloth systems out there work, by interconnecting a cloud of points with a network of springs, so why shouldn't you do just as much with your own setup?

The first and simplest use of such setups is something similar to an antenna or a feeler. Set up a null or a cube, which is going to be our root, with four children evenly spaced poles around it, and offset those on the Y-axis as in **Figure 4**.

Create an object for our end point in the middle of those four pins, make the pins passive rigid bodies, the end point an active rigid body, and leave the root as a normal object, then start a spring from each pin to the end point.

Here's your first springs network to play with. But... There's still one thing missing. When moving the root, and consequently the four pins, the end point is simply reacting to the springs, making the motion unlike what you'd probably want, as the end point will actually "precede" the root, while you would really want it to be delayed. That is simply solved by adding a drag force to the environment, so that the end point will have its motion delayed and diminished by the drag before it follows the root.

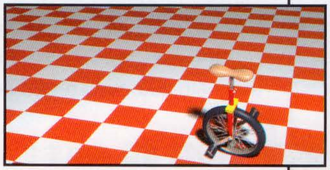
It's almost there, but there are still a couple points that need addressing:

First, our motion and the resistance of the network are fine, but only when planar–if you were to rotate the root point, results wouldn't be what you'd expect from anything even moderately stiff, and that is because right now our network is more similar to cloth than it is to a taut element. This is also easily addressed by reinforcing the springs network on the remaining axis as well, creating another two "vertical" pins (or using the root itself as one) and using springs between those and the end point as well (**figure 5**). Your final result should look roughly like scene: 04_RBDspringNetwork.scn.

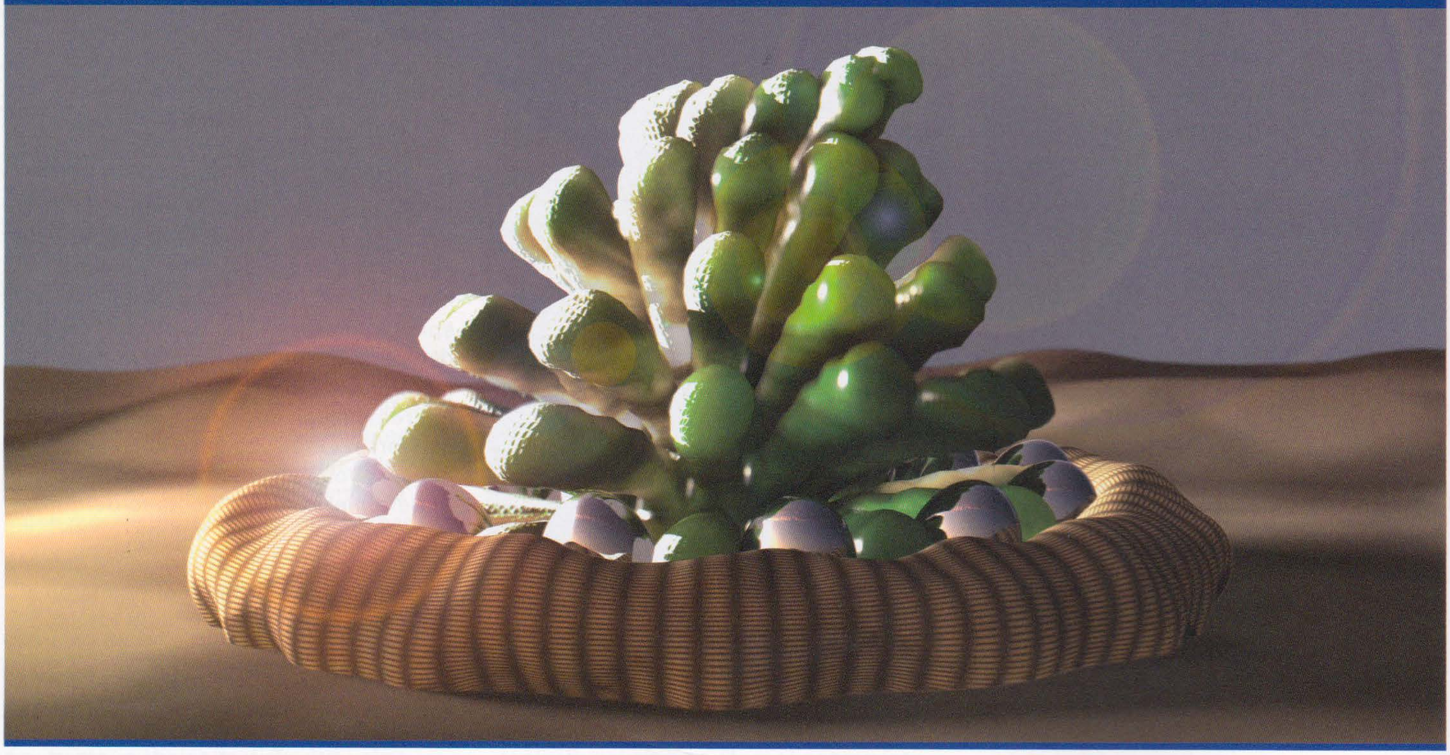
From here you have several options to expand on the subject, tweaking simple setups like this, making the network more taut or looser by strengthening the springs, more or less responsive by the dampening, and, most important, using hierarchies and expressions/constraints to create much larger and more complex networks. Once you have such a network you can use, by constraints or deformers, such objects to deform geometry (be careful, though, when rotations are produced), or more safely drive the positions alone of points on a curve or a mesh that can then be used as a deformer.

It takes a good deal of experimenting, some patience, and the basic concepts that hopefully I got across, but sets of constraints and dynamic objects are largely more high performance than people suspect. (I've had networks with over a hundred constraints and fairly large amounts of objects driving points of a cage at more than 5fps.) They also allow consistent and reproducible results, unlike quickstretch or simple scops that don't store the state of the objects. Last but not least, they offer full collisions and interaction capabilities, which is more than most hacks, or "creative solutions" if you're trying to sell your boss on it, provide.

***Raffaele Fragapane** is currently the Lead Character TD at Rising Sun Pictures in Adelaide, Australia. His background includes 10 years of industry employment in roles spanning from technical lead to on-set VFX coordination for commercials and films.*



AUTOMATING COMPOSITING AND RENDERING TASKS WITH NUKE



By Boaz Livny
3D Artist, Animator, Professor
New York, New York

Welcome back! This article follows up on the previous article "Compositing Passes with Nuke" in *HDRI 3D* Issue 6. In the previous article I introduced Nuke and its unique abilities for compositing and exporting multi-channel images. That article examined techniques for compositing different passes, as well as managing and creating multi-channel workflows, which is achieved by appending passes into their own unique channels within the comp downstream. Eventually those channels (opposed to separate images) are composited using different merge operations.

This article looks at enhancing our ability to work intuitively by nesting flowcharts within a single node, a group node. We then develop an effective user interface for that grouped node, which provides access to attributes within the nested flowchart. This sort of interface design for nested composites provides a means for controlling different knobs (attributes) for that flowchart, and all from a single grouped node. This process provides several workflow improvements that help automate repetitive tasks, such as color grading, color space conversions, format conversions, outputting several formats such as HD, video, CD, and web, custom optical effects, color correction processing, compiling and exporting OpenEXR images, compositing passes, and much more.
To clarify, the underlying idea is that you determine key requirements for your project, then create and group the flowchart, and finally export it as a gizmo. Creating gizmos allows us to export these nested flowcharts from their group nodes as processes that can then be reused by other compositors, without their ability to interact or "damage" the internal flowchart. Essentially, a gizmo provides access to a group nodes interface and settings, hiding the flowchart from the user. This process can efficiently prevent a need to repeatedly create the same flowchart within scripts, as well as provide a cleaner workspace by hiding embedded nodes from the user. Hence, essentially gizmos are composites that are grouped under a node that can be exported and imported into other scripts. The embedded flowchart is then hidden from the user, within that gizmo. For controlling these nodes, during the creation process, the creator must specify specific attributes that are added to the group (Gizmo's) UI. After exporting the gizmo, only those attributes will be available to users. For lead compositors or technical directors, this is a way to define specific settings while concealing others from users, as once a gizmo is exported, its internal flowchart is hidden from the user. Also, gizmos act much like instanced nodes, the TD can make corrections or revisions to a gizmo, which then affects other scripts that are using the gizmo. To clarify, once a gizmo is imported, it still maintains a relationship with its external script, thus, if the gizmo changes, updates will affect the entire production. This sort of characteristic only applies to exported gizmos, and not to internal grouped nodes.

Nuke provides a powerful set of tools to custom create any sort of UI for these gizmos, without a need to have any programming knowledge—hence, a more artist friendly approach. You may think of this as a means for creating custom nodes and operations by creatively combining existing ones as well as using TCL expressions (if you're comfortable with scripting). One other requirement for piping gizmos into flowcharts is that they must posses at minimum one output, and may also provide inputs. Hence, you may specify any number of inputs and one output for a gizmo. This topic will be clarified with the examples below. Another great advantage with gizmos is that once exported, they can be placed within Nuke's plugin directory and loaded into the Nuke menu as additional menu items, similar to plugins. This is achieved by adding a gizmo description within the Nuke menu.tcl file, providing a way for loading the gizmo

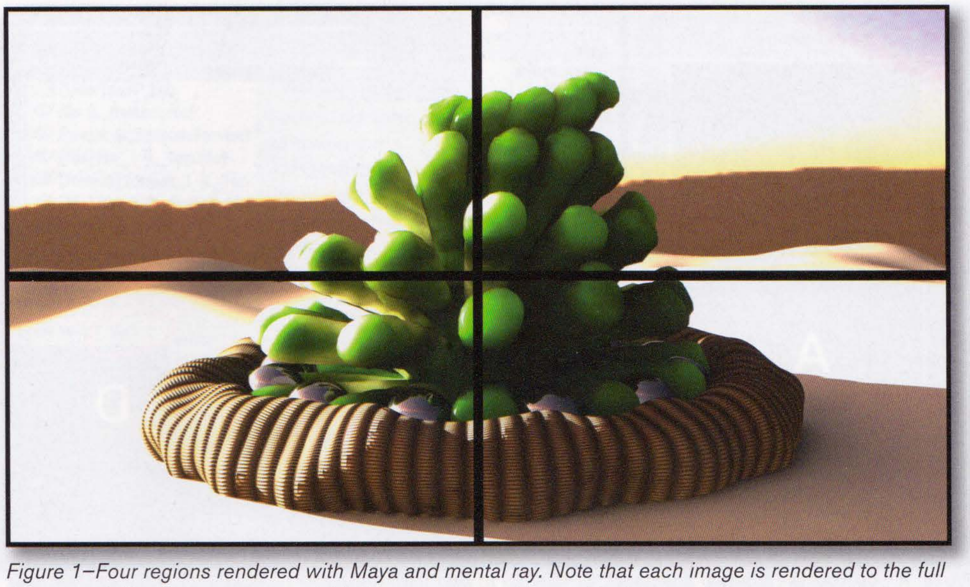


Figure 1–Four regions rendered with Maya and mental ray. Note that each image is rendered to the full frame resolution; hence, areas outside the rendered region are black for each image, respectively.

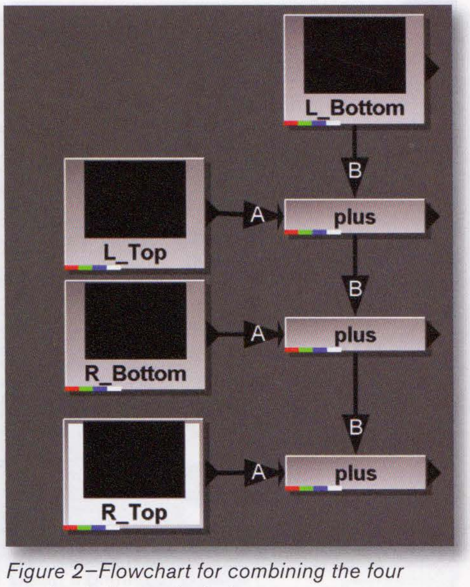


Figure 2–Flowchart for combining the four quadrants into a single frame.

just as any other node, as well as defining a hotkey. For an extensive production, the fact a compositor can simply press a hotkey to load a custom made flowchart is obviously a great advantage. You can find more information on the required syntax for defining gizmos as menu items within the Nuke User Guide, which is available as a PDF file when you download a trial. Note: Gizmos can simply be imported into a script rather than added as menu items, and hence do not require the latter to function properly.

The first tutorial in this article examines a basic gizmo, introducing key tools and workflows. We will create a gizmo that automates the task of combining a single image from four separate images, seen in **Figure 1**. This means we must provide four inputs as well as one output that typically connects into a Write node. The inputs may take the form of actual Read nodes that are connected into the gizmo, or alternately these Read nodes may be embedded within the gizmo. The latter means that images are then loaded directly into the gizmo through its interface rather than Read nodes connecting into the gizmo. In the first tutorial we look at this process of embedding a Read node within the gizmo. The second example examines the process of developing a gizmo for compositing subsurface scattering, which directly relates to my previous article on subsurface scattering in the mental ray for Maya series, *HDRI 3D* Issue 8. In this example we look at using input nodes that receive external Read nodes, or any other node, as input.

CREATING A GIZMO FOR COMPILING RENDERED REGIONS

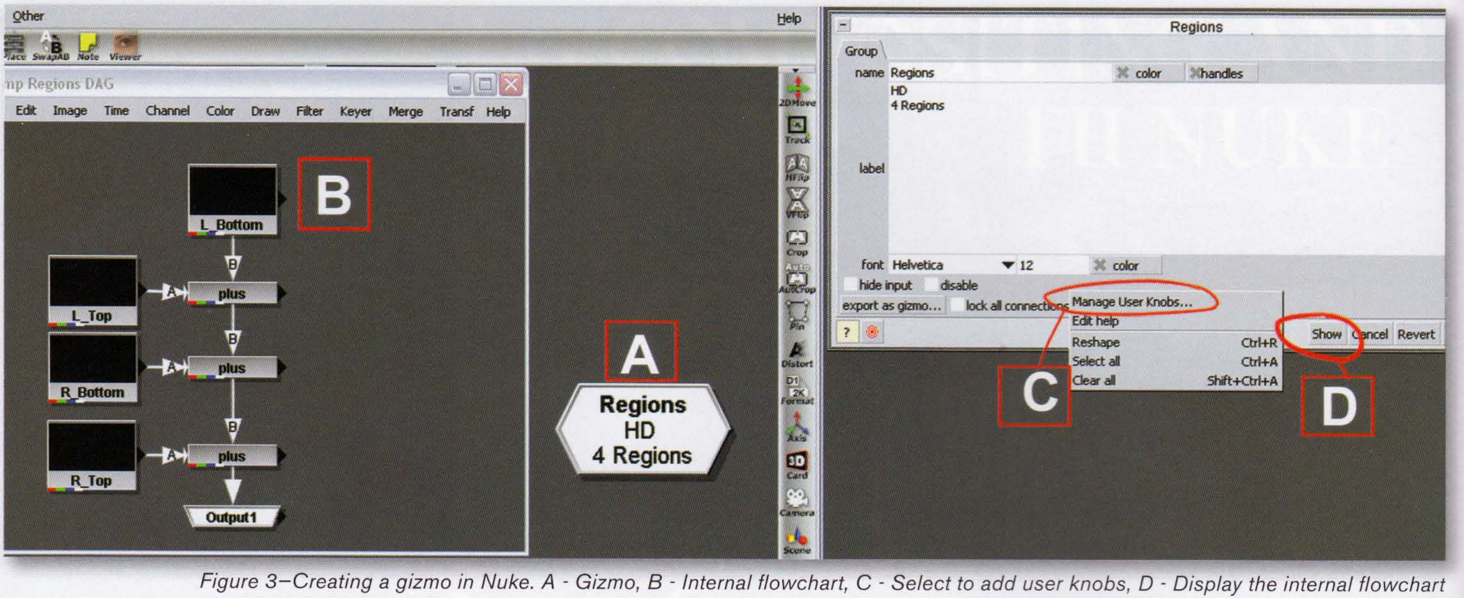
If you ever had to deal with a particularly heavy scene that may bring a render to a crawl, or may cause the render to abort, you may have considered rendering regions. Several render engines support rendering into regions, which is the process of dividing a frame to any number of regions, each of which is then rendered as a separate image. Below is an example of Maya using a command line render to render with mental ray an HD frame into four regions. Typically this should be typed into a single file that can be executed, rather than executing one at a time from the command line. Simply append the .BAT suffix on Windows systems to make an executable file, or use the Chmod command on Linux/OSX systems. When executed it will render four images representing the four quadrants that combine in the full image.

```
Commandline example:

render -r mr -reg 0 960 0 540 -im
L_Bottom -s 1 -e 1 -v 5 S:\Maya\
projects\default\scenes\regions.mb

render -r mr -reg 0 960 540 1080
-im L_Top -s 1 -e 1 -v 5 S:\Maya\
projects\default\scenes\regions.mb

render -r mr -reg 960 1920 0 540 -
im R_Bottom -s 1 -e 1 -v 5 S:\Maya\
projects\default\scenes\regions.mb

render -r mr -reg 960 1920 540 1080
-im R_Top -s 1 -e 1 -v 5 S:\Maya\
projects\default\scenes\regions.mb
```

Processing these types of images into a single image can become tedious, especially if using this often. Obviously this is not the most creative process. Using Nuke we create a gizmo that processes these images into a single frame, which then can either be rendered (recommended) or used as a base for the comp flowchart. This example, as mentioned earlier, will include embedding the read nodes within the gizmo. You can download the image files and Nuke files from the Resource page on the *HDRI 3D* Web site.

STEP 1: Create four Read nodes and label them appropriately under the Node tab name attribute for each Read node. Use unique naming conventions, as seen in **Figure 2**. Set the default format for the images under the Read node Read tab, and leave File blank. In this case we will create an HD format; however, we will also provide the user with a way to change image resolution (format). Remember: Once a gizmo is exported, the internal settings will be hidden from the user; hence, we must consider all the required settings now, while creating the gizmo.

STEP 2: Create the appropriate Merge nodes (M hotkey) that combine the four images together. Set Operation to Plus under the merge node tab, and enable all four channels (RGBA) for both input A channels and B Channels. Connect the Merge nodes as seen in **Figure 2**, so that each Read node is added into the flowchart. Since these images are all in the same resolution, the black area outside each image's rendered region overlaps color areas of other images, as implied in **Figure 1**. This means using plus adds color values from each image onto black (zero); hence, when combined it provides for an exact match without any color change. Note


Other
Help
Place SwapAB Note Viewer
mp Regions DAG
Edit Image Time Channel Color Draw Filter Keyer Merge Transf Help
L_Bottom
B
L_Top
plus
R_Bottom
plus
R_Top
plus
Output1
A
Regions
HD
4 Regions
2DMove Track HFlip VFlip Crop AutoCrop Pin Distort 2K Format Axis Card Camera Scene
Regions
Group
Manage User Knobs...
Edit help
Reshape Ctrl+R
Select all Ctrl+A
Clear all Shift+Ctrl+A
Show Cancel Revert
C
D

name Regions
label HD 4 Regions
font Helvetica 12
☐ hide input ☐ disable
export as gizmo... ☐ lock all connections

Figure 3–Creating a gizmo in Nuke. A - Gizmo, B - Internal flowchart, C - Select to add user knobs, D - Display the internal flowchart (only available within gizmo creation scripts).

that if areas outside the rendered region are other then black, the plus operation would not provide a solution, as it would be influenced by that color value.

STEP 3: For creating the gizmo, group the nodes by selecting all the nodes, then under the Other menu select Group, or press the Cntrl + G shortcut. The grouped nodes are placed within a separate window (**figure 3**, B) that is actually revealing the gizmo's internal nodes. Notice these nodes are copied into the gizmo as they are still available within the main script window. These nodes can now be deleted since we no longer require them. The A mark in **Figure 3** shows the gizmo after it has been named appropriately under its Group tab setting, also seen in **Figure 3** on the right.

STEP 4: Notice that the gizmo automatically appended an Output node under the last Merge (plus) node. Normally Nuke will "fill in the blanks," meaning if you provide nodes that expect inputs, Nuke automatically adds Input nodes, as well as an Output node. These Input and Output nodes simply enable the black arrows we use to pull connections from one node to another, meaning they provide a way to drive connections in and out from the gizmo. In case you do not see an Output node, manually create one from under the Other menu within the Gizmo. Connect the Output node to the bottom Merge node, which in return enables connecting a gizmo to a Write node, as seen in **Figure 3** on the left (under the area marked B).

Note that during the creation process, if you close the gizmo window it can be reopened under the Gizmo's settings tab by selecting Show, marked with a D in **Figure 3**.

STEP 5: Once the gizmo is exported, the internal flowchart will be hidden from other users, and thus we must provide access to the Read nodes File and Format settings. With the cursor placed over the Gizmo's Group tab, right-click to reveal the drop-down menu, as seen in **Figure 3** marked C. Select Manage User Knobs... to reveal a window that enables adding user knobs to a gizmo, as seen in **Figure 4** on the left.

The Manage User Knobs window enables adding knobs to the gizmo's user tabs from within the flowchart, as well as creating custom knobs. Additional custom UI settings (from under the Add... option) can be anything from floating point sliders, static text, divider lines, drop-down lists, and additional tabs, to 3D positing vectors, to mention but a few. Select Pick from the Manage User Knob window, which in turn opens another window, also seen in **Figure 4,** on the right. This window enables us to browse to any operator within the flowchart and add it to the gizmo's user tab. Note the gizmo's user tab does not exist until you add at least one node; once added, you will see an additional tab appear within the gizmo's settings. Browse to the Read nodes as seen in **Figure 4** and select file and format for each of the four Read nodes–you can hold the control button to select more than one attribute at a time. Once all four file and format attributes for all four Read nodes have been selected, press OK to complete this operation. The selected knobs are added into the list, as seen in **Figure 5**.

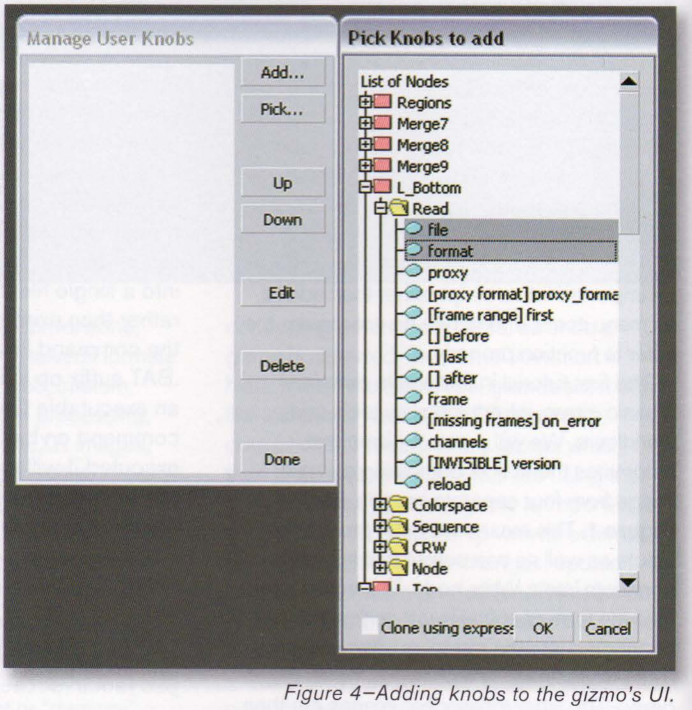
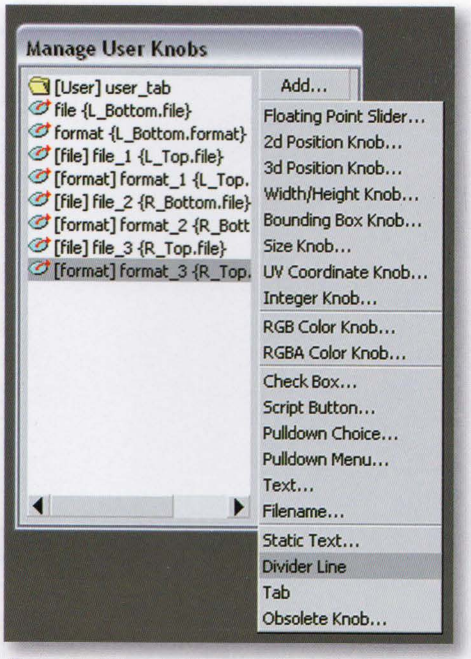

Manage User Knobs
Add...
Pick...
Up
Down
Edit
Delete
Done
Pick Knobs to add
List of Nodes
Regions
Merge7
Merge8
Merge9
L_Bottom
Read
file
format
proxy
[proxy format] proxy_forma
[frame range] first
[] before
[] last
[] after
frame
[missing frames] on_error
channels
[INVISIBLE] version
reload
Colorspace
Sequence
CRW
Node
L_Top
☐ Clone using expres:
OK
Cancel

Figure 4–Adding knobs to the gizmo's UI.

STEP 6: For better organization in the gizmo's user tab, we add divider lines between the different knobs as well as label knobs appropriately so they make sense to users. Select Add... from under the Manage User Knobs window, as seen in **Figure 5** on the right. From the drop-down list add Divider Line four times, then use the Up

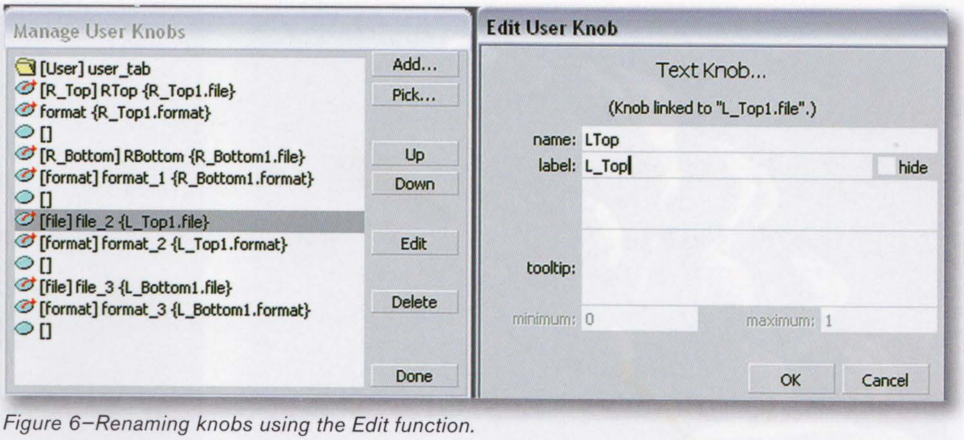


Manage User Knobs

[User] user_tab
file {L_Bottom.file}
format {L_Bottom.format}
[file] file_1 {L_Top.file}
[format] format_1 {L_Top.
[file] file_2 {R_Bottom.file}
[format] format_2 {R_Bott
[file] file_3 {R_Top.file}
[format] format_3 {R_Top.

Add...
Floating Point Slider...
2d Position Knob...
3d Position Knob...
Width/Height Knob...
Bounding Box Knob...
Size Knob...
UV Coordinate Knob...
Integer Knob...
RGB Color Knob...
RGBA Color Knob...
Check Box...
Script Button...
Pulldown Choice...
Pulldown Menu...
Text...
Filename...
Static Text...
Divider Line
Tab
Obsolete Knob...

Figure 5–After picking specific knobs from within the flowchart, they are added into the window and can also be seen under the User tab for that gizmo.



Manage User Knobs

[User] user_tab
[R_Top] RTop {R_Top1.file}
format {R_Top1.format}
[]
[R_Bottom] RBottom {R_Bottom1.file}
[format] format_1 {R_Bottom1.format}
[]
[file] file_2 {L_Top1.file}
[format] format_2 {L_Top1.format}
[]
[file] file_3 {L_Bottom1.file}
[format] format_3 {L_Bottom1.format}
[]

Edit User Knob

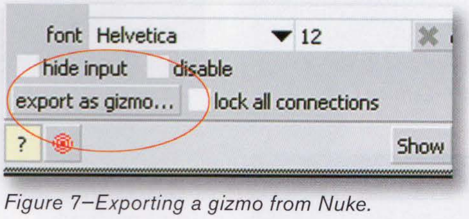
Text Knob...

(Knob linked to "L_Top1.file".)

name: LTop
label: L_Top ☐ hide
tooltip:
minimum: 0 maximum: 1

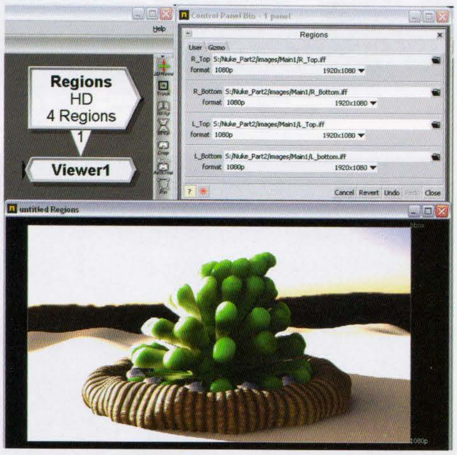
Add...
Pick...
Up
Down
Edit
Delete
Done
OK
Cancel

Figure 6–Renaming knobs using the Edit function.



font Helvetica 12
☐ hide input ☐ disable
export as gizmo... ☐ lock all connections

Figure 7–Exporting a gizmo from Nuke.



Regions
HD
4 Regions
1
Viewer1
Regions
untitled Regions

Figure 8–Regions gizmo imported and tested in a new script.

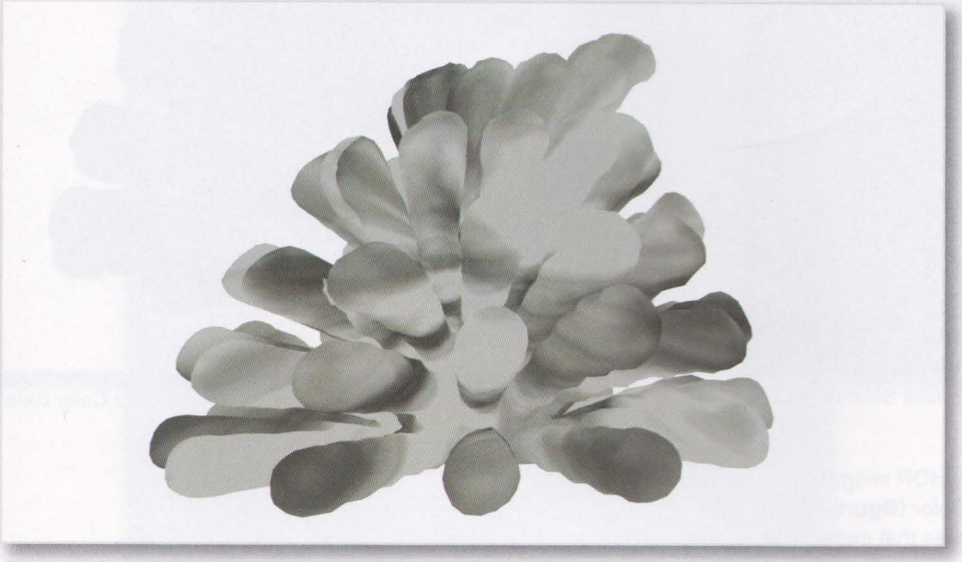


Figure 9–Overall Color pass.

and Down buttons to spread the divider lines between the file and format knobs to separate Read nodes with a line, as seen in **Figure 6** on the left. Label the File knobs appropriately by selecting them one at a time from the list and press the Edit button (**figure 6** on the right). The Name setting is used to provide each knob with a unique name so that it can be accessed with TCL or some sort of knob linking. I recommend you provide unique names to prevent any chance of knob names conflicting. The Label is used to provide a name that will be displayed for that knob in the gizmo's user tab, and hence is the label users will see while compositing. Name each file and format appropriately, as seen in **Figure 6** on the right. I named the file knobs using L_Top, L_Bottom, R_Top, and R_Bottom to identify the four quadrants and I left format as format since the divider lines will help associate each format with its file knob respectively, as in **Figure 6**.

STEP 7: To finalize and export the gizmo, we simply select Export As Gizmo under the gizmo's group tab seen in **Figure 7**. If you make sure the gizmo has a *.gizmo* extension, the gizmo can be added into Nuke's menu by placing it in the plugin directory and adding it to the menu.tcl file, as mentioned earlier.

Once you have completed preparing the gizmo, you can test it by creating a new script and importing the gizmo. Open its user tab and locate the four image files, set the format, and connect a viewer, as seen in **Figure 8**. At this stage your gizmo should be working properly. Now that you have a gizmo for regions, you may consider more often rendering regions if rendering becomes excessively heavy, as combining these images is as easy as loading the gizmo, connecting a write node, and executing a render.

CREATING A GIZMO FOR SUBSURFACE SCATTERING

In the following section I review an example for creating a gizmo that composites subsurface scattering (sss). First I determine key features I will need, including the passes I most likely will use for compositing subsurface scattering. These considerations are based on the fact that I'm rendering these passes with mental ray for Maya. The passes I included are Overall Color (**figure 9**), Diffuse Color

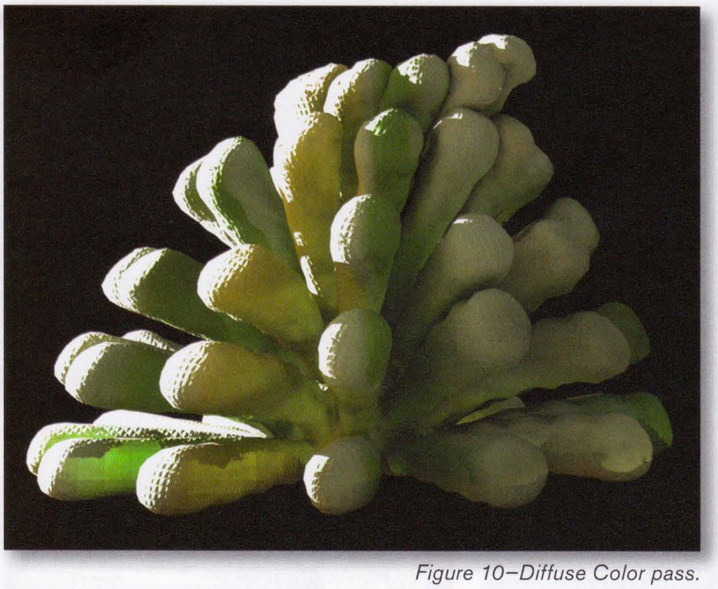


Figure 10–Diffuse Color pass.

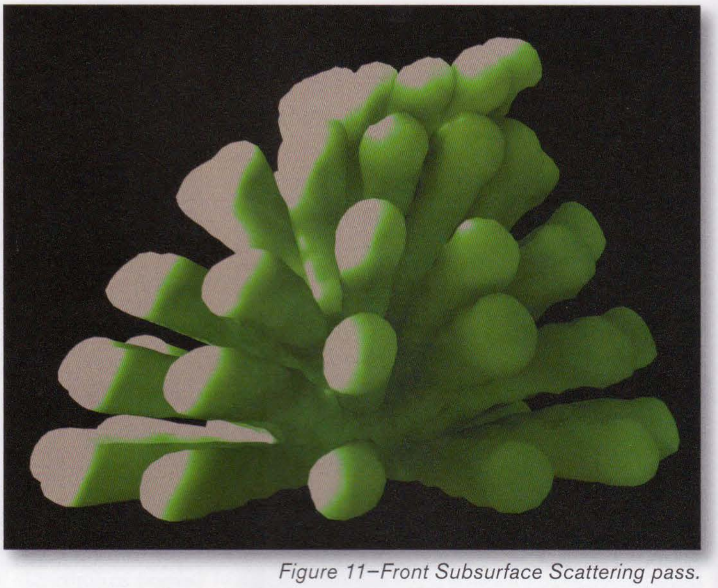


Figure 11–Front Subsurface Scattering pass.

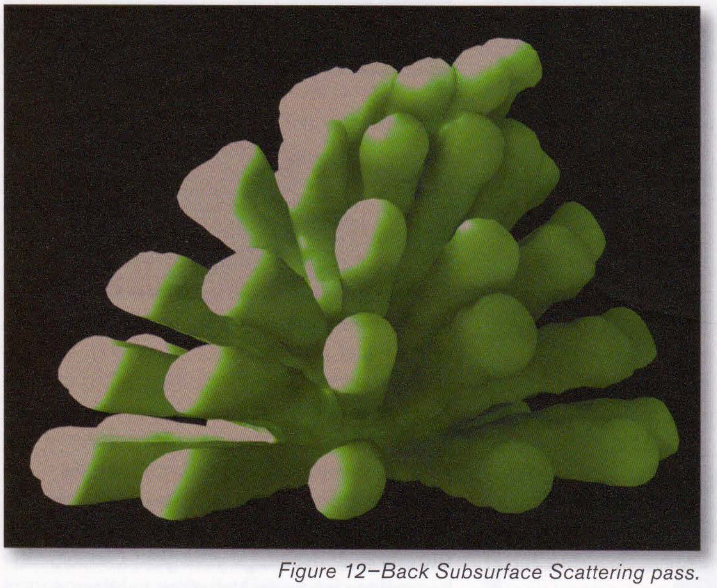


Figure 12–Back Subsurface Scattering pass.

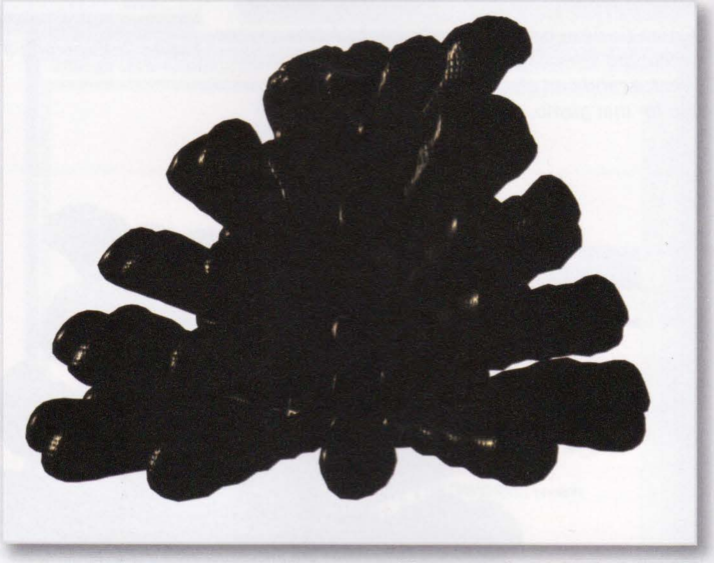
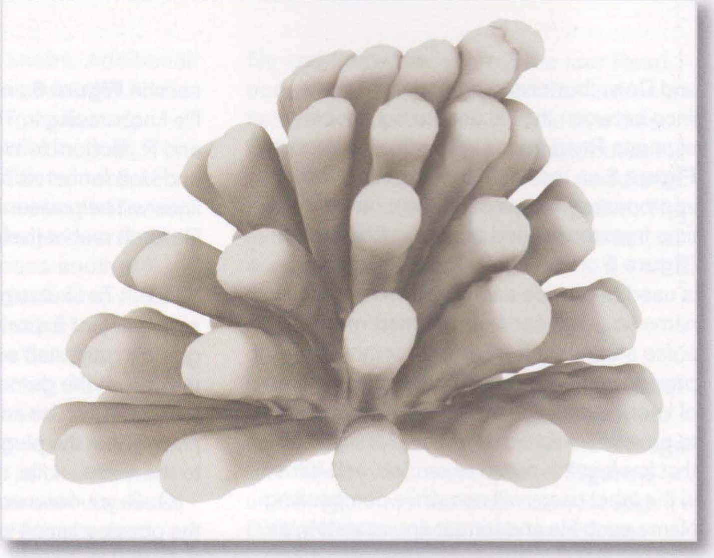


Figure 13–Specular Color pass.

(**figure 10** -HDR image), Front SSS (**figure 11** -HDR image), Back SSS (**figure 12** -HDR image), Specular Color (**figure 13**), Ambient Occlusion (**figure 14**), and an Alpha pass that can be used to help composite the gizmo into an external flowchart. The basic compositing process for these passes is seen in **Figure 15**.

I am basing the compositing process used to create this flowchart on mental ray's non-physical subsurface phenomenon shader. For more information on this shader and how it is evaluated, see the previous sss article in the mental ray for Maya series, *HDRI 3D* Issue 8. For creating the flowchart we must consider how sss is calculated at render time. Based on that process, I created a similar process that can be seen in **Figure 15**. The sss phenomenon shader allows defining diffuse and specular colors as well as additional subsurface scattering layers, which are Front Subsurface Scattering (Front SSS) and Back Subsurface Scattering (Back SSS). The Diffuse, Front SSS, and Back SSS are all added using screen compositing to avoid overexposure. **Figure 15** (marked A) shows a merge screen operation used to combine the front sss with back sss; then, using an additional merge screen operation, that resulting color is merged with the diffuse pass. In **Figure 16** the node labeled sss_diffuse shows the settings used for merging the diffuse color pass with the subsurface merge node. The Overall color pass is then used as a multiplier against the total colors added from the first two operations

Figure 14–Ambient Occlusion pass.



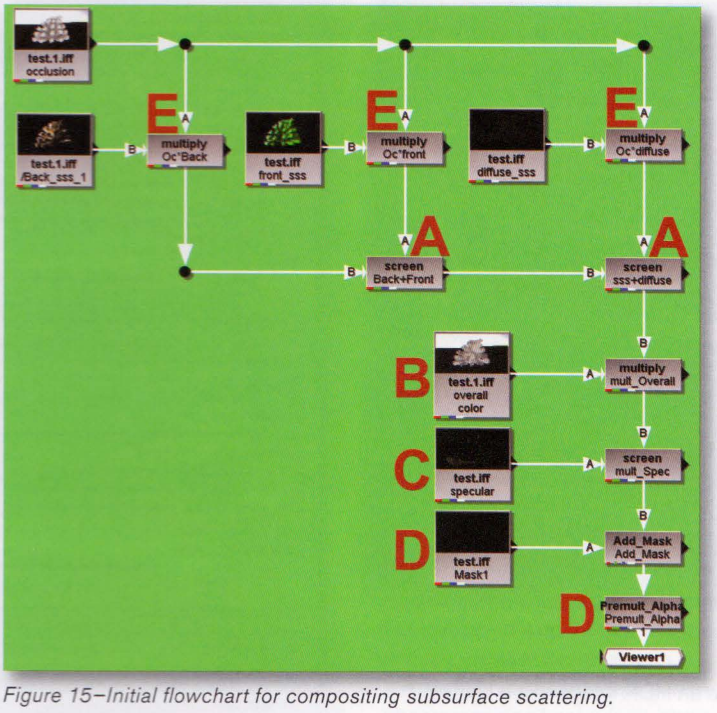
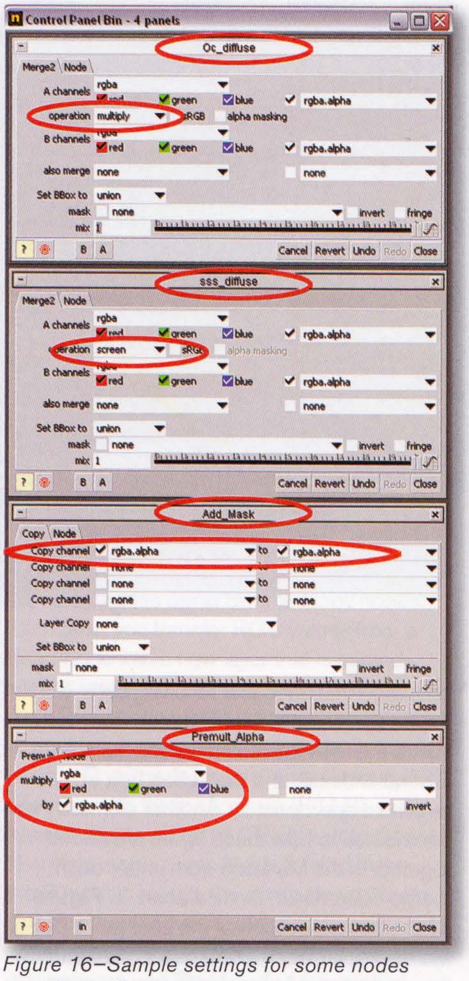

test.1.iff
occlusion
test.1.iff
/Back_sss_1
E
multiply
Oc*Back
test.iff
front_sss
E
multiply
Oc*front
test.iff
diffuse_sss
E
multiply
Oc*diffuse
A
screen
Back+Front
A
screen
sss+diffuse
B
test.1.iff
overall
color
multiply
mult_Overall
C
test.iff
specular
screen
mult_Spec
D
test.iff
Mask1
Add_Mask
Add_Mask
D
Premult_Alpha
Premult_Alpha
Viewer1

Figure 15–Initial flowchart for compositing subsurface scattering.


Control Panel Bin - 4 panels

Oc_diffuse

Merge2 | Node

A channels: rgba ☑ red ☑ green ☑ blue ✓ rgba.alpha
operation: multiply ☐ sRGB ☐ alpha masking
B channels: rgba ☑ red ☑ green ☑ blue ✓ rgba.alpha
also merge: none ☐ none
Set BBox to: union
mask: ☐ none ☐ invert ☐ fringe
mix: 1

sss_diffuse

Merge2 | Node

A channels: rgba ☑ red ☑ green ☑ blue ✓ rgba.alpha
operation: screen ☐ sRGB ☐ alpha masking
B channels: rgba ☑ red ☑ green ☑ blue ✓ rgba.alpha
also merge: none ☐ none
Set BBox to: union
mask: ☐ none ☐ invert ☐ fringe
mix: 1

Add_Mask

Copy | Node

Copy channel: ✓ rgba.alpha to ✓ rgba.alpha
Copy channel: ☐ none to ☐ none
Copy channel: ☐ none to ☐ none
Copy channel: ☐ none to ☐ none
Layer Copy: none
Set BBox to: union
mask: ☐ none ☐ invert ☐ fringe
mix: 1

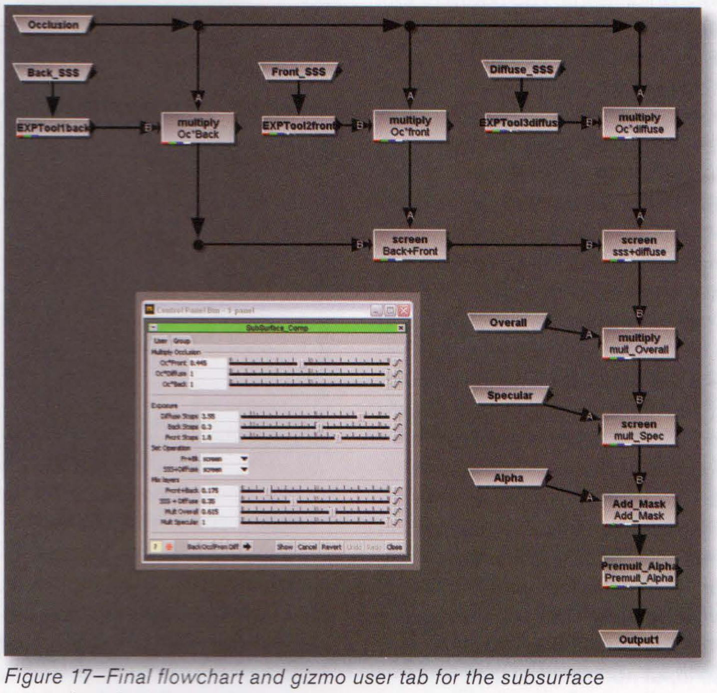
Premult_Alpha

Premult | Node

multiply: rgba ☑ red ☑ green ☑ blue ☐ none
by: ✓ rgba.alpha ☐ invert

Cancel
Revert
Undo
Redo
Close

Figure 16–Sample settings for some nodes used in the flowchart.


Occlusion
Back_SSS
EXPTool1back
multiply
Oc*Back
Front_SSS
EXPTool2front
multiply
Oc*front
Diffuse_SSS
EXPTool3diffus
multiply
Oc*diffuse
screen
Back+Front
screen
sss+diffuse
Overall
multiply
mult_Overall
Specular
screen
mult_Spec
Alpha
Add_Mask
Add_Mask
Premult_Alpha
Premult_Alpha
Output1

Control Panel Bin - 1 panel

SubSurface_Comp

User | Group

Multiply Occlusion
Oc*Front: 0.445
Oc*Diffuse: 1
Oc*Back: 1

Exposure
Diffuse Stops: 3.55
Back Stops: 0.3
Front Stops: 1.8

Set Operation
Fr+Bk: screen
SSS+Diffuse: screen

Mix layers
Front+Back: 0.175
SSS + Diffuse: 0.35
Mult Overall: 0.615
Mult Specular: 1

Show
Cancel
Revert
Undo
Redo
Close

Figure 17–Final flowchart and gizmo user tab for the subsurface scattering gizmo.

(marked B). Specular color (marked C) is then added onto the resulting color value. The last two connections (marked D) add an alpha channel using the Copy Channels operator (**figure 16**, labeled Add_Mask) into the flowchart. Then using a Premult node (**figure 16**, labeled Premult_Alpha) I multiply the flowchart's RGB values against the alpha channel, a common compositing procedure.

One of my main objectives is to enable tweaking subsurface scattering just as I would in Maya. Hence, this gizmo actually is meant to provide a much faster workflow than that of re-rendering repeatedly after every change. This would explain some of the passes and operators I applied in the flowchart. I decided to add an additional occlusion pass to the workflow, which is multiplied against each of the color contributors (**figure 15**, marked E). **Figure 16** labeled Oc_diffuse shows an example for settings for one of these merge multiply nodes that is used to multiply the occlusion pass with the diffuse pass (rgba.alpha should have been disabled). I prefer to multiply the occlusion pass separately, with each pass providing more control over its influence on each pass. Using the Mix slider from under each merge multiply node (for occlusion), I control the amount of influence occlusion will have on each pass respectively. Obviously, this means each of those sliders are added into the gizmo user tab, as seen in **Figure 17** under Multiply Occlusion. To add such labeling text (Multiply Occlusion) into the user tab, simply select Add... from under the Manage User Knobs window, then select Static Text, and label an entry. To add sliders I used the Pick option and located the Mix knobs for each of the occlusion merge multiply nodes, then renamed them appropriately using the Edit function.

I decided to provide an option to change the type of merge operation used for screen compositing the passes. In other words, I set the default merge operation to screen compositing (as within Maya), but also provided the user with an option for changing this operation, for example, to a Plus operation. This option is seen in **Figure 17** under Set Operation, which allows the user to change the merge operation used to add both front sss and back sss (Fr+Bk) as well as change the

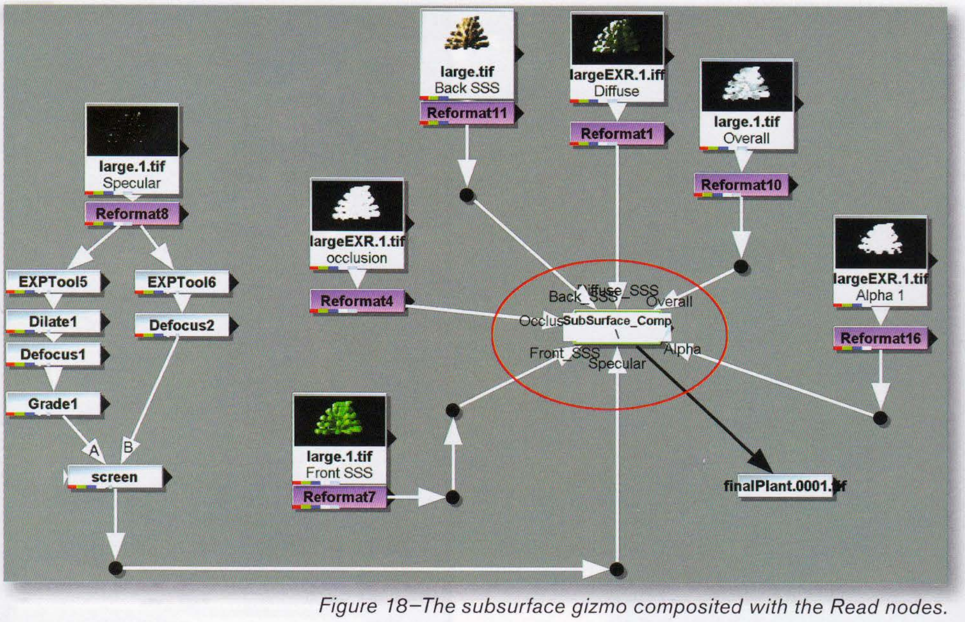


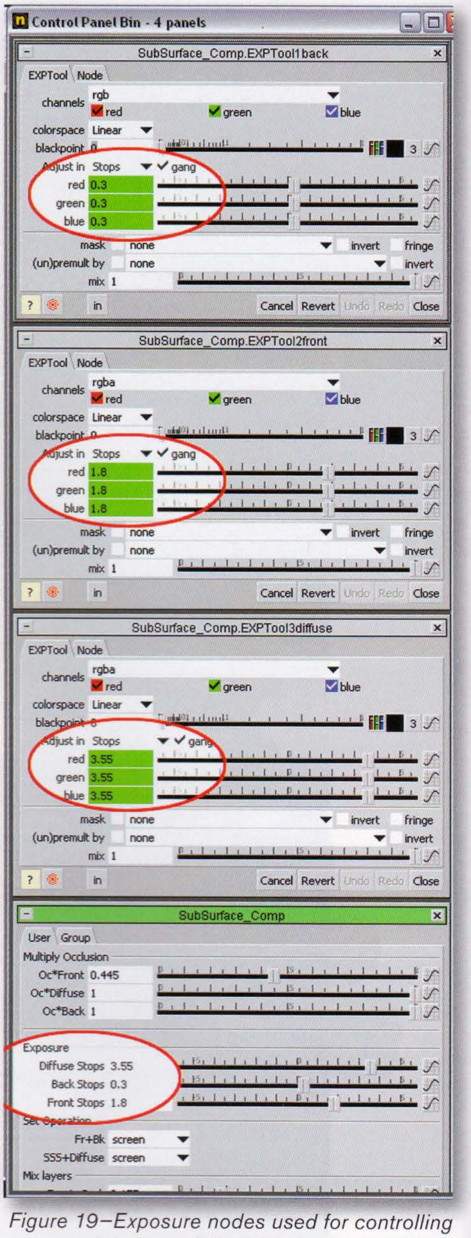
Figure 18–The subsurface gizmo composited with the Read nodes.

operation that adds that resulting color with the diffuse color (SSS+Diffuse). Obviously, I used the Pick function to locate the merge Operation knob, and then used Edit to rename it appropriately. Another element that contributes to how these layers are added together is the Mix knob from under each merge operation in the flowchart. In **Figure 17** the bottom section of the user tab, labeled Mix layers, provides a means for changing the amount of influence each merge node applies. You can see that I have control over the influence between front and back sss passes, the sss layers and the diffuse pass, the overall color multiply intensity, and the specular intensity.

One other factor that greatly influences sss is light intensity. To provide more control over the light intensity, mainly over the front and back scatter layers, I rendered HDR images using the OpenEXR format (RGBE). This provides a broader range of color for describing these passes. One interesting point of consideration with respect to rendering passes is that normally we separate light passes from color passes, then composite them together, a process described in detail in the previous Nuke article in *HDRI 3D* Issue 6. Having said that, we are now faced with a dilemma, since if we render color passes with no light, or more accurately a constant light, then front and back sss passes will not render properly. SSS passes are primarily based on light intensities and directionality. A workaround would be to render the front and back sss passes using lighting, then render the diffuse pass with a constant light (surface shader, or ambient light with no shading), just as you would render a standard diffuse pass. The diffuse pass can then be multiplied with a light pass before being piped into the gizmo. With respect to this example, the diffuse pass was also rendered as an HDR image using lighting.

This brings me to yet another point of consideration, which is that I do not want to include Read nodes within this gizmo. This means Read nodes may go through some preprocessing before being piped into the gizmo. Hence, I prefer to use Input nodes (found under *Other > Input*), providing inputs to the various merge nodes in the flowchart. **Figure 17** shows that all the Read nodes from the initial flowchart have been removed and changed into Input nodes; obviously, all are named appropriately from under their Input tab Name setting. In **Figure 18** the area circled in red demonstrates the importance of properly naming Input nodes, as those names are displayed when input connectors (the black arrows) are pulled out from the gizmo. We also see that the specular pass has gone through some preprocessing to create a specular bloom as well as fine-tune the specular pass. Once the specular flowchart is finalized, it is added into the sss gizmo using the specular input. This demonstrates yet another reason I prefer to use Input nodes rather than Read nodes within the gizmo. **Figure 18** also shows an Output node connected to a Write node for executing a render.

For this example, as mentioned earlier, I rendered HDR images for all three color passes (diffuse, and front and back sss) using a key light. As I decided to use HDR images, I also added an Exposure node for each of these nodes (diffuse, and front and back sss), providing more control over exposure levels, as seen in **Figure 17**. This means if you do render HDR images, you will have more control over the light intensity during the compositing stage. Also, note that if you don't use HDR, you still will have another setting that may prove handy when tweaking color values for these passes. **Figure 19** shows the three exposure node settings and the gizmo settings (bottom). Notice that each exposure node has three sliders that control the RGB channels exposure separately, as well as some other options.


Control Panel Bin - 4 panels

SubSurface_Comp.EXPTool1back
EXPTool Node
channels rgb ☑ red ☑ green ☑ blue
colorspace Linear
blackpoint 0
Adjust in Stops ✓ gang
red 0.3
green 0.3
blue 0.3
mask ☐ none ☐ invert ☐ fringe
(un)premult by ☐ none ☐ invert
mix 1

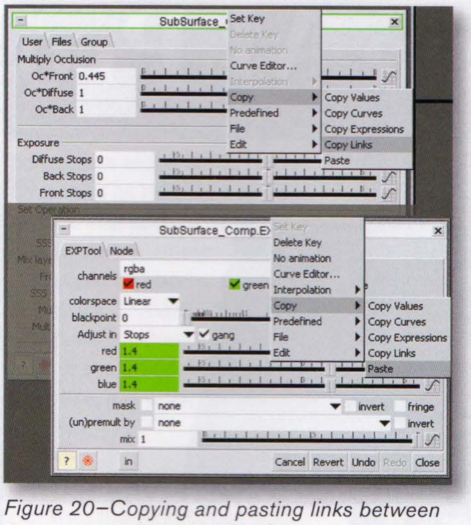
SubSurface_Comp.EXPTool2front
EXPTool Node
channels rgba ☑ red ☑ green ☑ blue
colorspace Linear
blackpoint 0
Adjust in Stops ✓ gang
red 1.8
green 1.8
blue 1.8
mask ☐ none ☐ invert ☐ fringe
(un)premult by ☐ none ☐ invert
mix 1

SubSurface_Comp.EXPTool3diffuse
EXPTool Node
channels rgba ☑ red ☑ green ☑ blue
colorspace Linear
blackpoint 0
Adjust in Stops ✓ gang
red 3.55
green 3.55
blue 3.55
mask ☐ none ☐ invert ☐ fringe
(un)premult by ☐ none ☐ invert
mix 1

SubSurface_Comp
User Group
Multiply Occlusion
Oc*Front 0.445
Oc*Diffuse 1
Oc*Back 1
Exposure
Diffuse Stops 3.55
Back Stops 0.3
Front Stops 1.8
Set Operation
Fr+Bk screen
SSS+Diffuse screen
Mix layers

Figure 19–Exposure nodes used for controlling the stops of the HDR images, and their respective sliders under the gizmo's user tab.

To make this process easier, I want a single slider to control each of these exposure tab's RGB stops. I achieve this by adding a Floating Point Slider with the Add... button in the Manage User Knobs window. After adding sliders to the gizmo, I label them appropriately and place them under an Exposure section in the gizmo user tab. For driving connections between these sliders and the sliders from the exposure nodes, I define a dependency link. This link will use one slider from under the gizmo to drive three sliders on the relevant exposure tab. Technically I achieve this by placing the cursor over the box to the right of each slider (gizmo sliders) and select Copy



Multiply Occlusion
Oc*Front 0.445
Oc*Diffuse 1
Oc*Back 1
Exposure
Diffuse Stops 0
Back Stops 0
Front Stops 0

channels rgba ☑ red ☑ green
colorspace Linear
blackpoint 0
Adjust in Stops ✓ gang
red 1.4
green 1.4
blue 1.4
mask ☐ none ☐ invert ☐ fringe
(un)premult by ☐ none ☐ invert
mix 1

Figure 20–Copying and pasting links between the exposure nodes' RGB sliders and the gizmo exposure sliders.

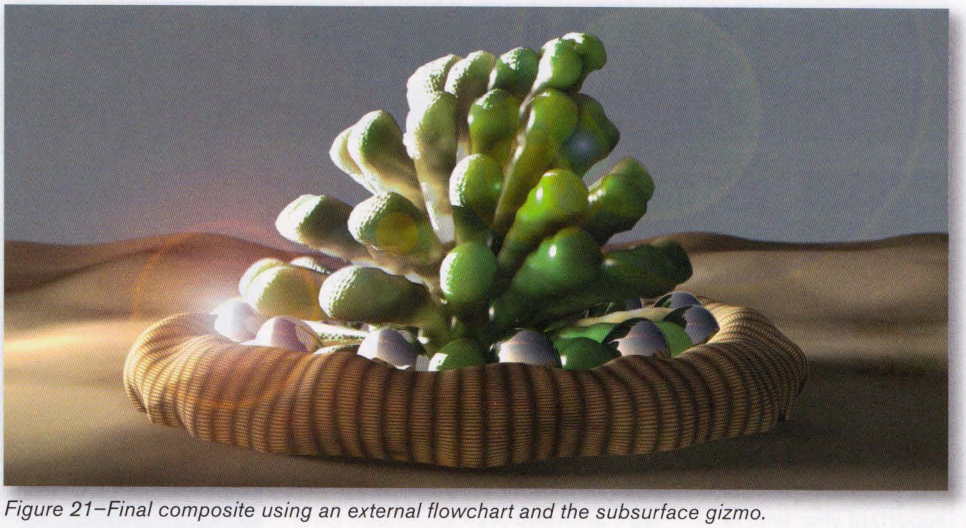
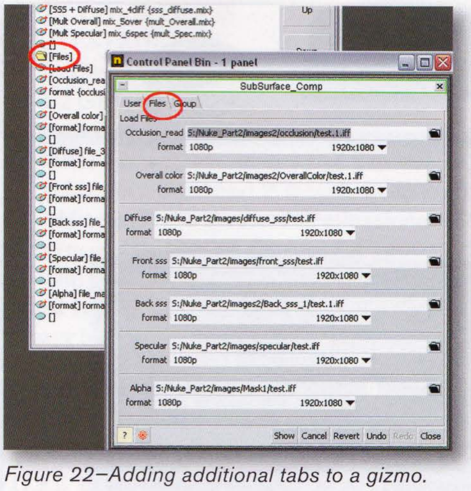


Figure 21–Final composite using an external flowchart and the subsurface gizmo.



[SSS + Diffuse] mix_4diff {sss_diffuse.mix}
[Mult Overall] mix_5over {mult_Overall.mix}
[Mult Specular] mix_6spec {mult_Spec.mix}
[]
[Files]
[Load Files]
[Occlusion_rea
format {occlusi
[]
[Overall color]
[format] forma
[]
[Diffuse] file_3
[format] forma
[]
[Front sss] file
[format] forma
[]
[Back sss] file_
[format] forma
[]
[Specular] file_
[format] forma
[]
[Alpha] file_ma
[format] forma
[]

Occlusion_read S:/Nuke_Part2/images2/occlusion/test.1.iff
format 1080p 1920x1080
Overall color S:/Nuke_Part2/images2/OverallColor/test.1.iff
format 1080p 1920x1080
Diffuse S:/Nuke_Part2/images/diffuse_sss/test.iff
format 1080p 1920x1080
Front sss S:/Nuke_Part2/images/front_sss/test.iff
format 1080p 1920x1080
Back sss S:/Nuke_Part2/images2/Back_sss_1/test.1.iff
format 1080p 1920x1080
Specular S:/Nuke_Part2/images/specular/test.iff
format 1080p 1920x1080
Alpha S:/Nuke_Part2/images/Mask1/test.iff
format 1080p 1920x1080

Figure 22–Adding additional tabs to a gizmo.

Links, as seen in **Figure 20** (top). Then I navigate to the respective exposure node (**figure 20** bottom) and right-click over the box to the right of each RGB channel, each time selecting *Copy > Paste* (three times). This will define a dependency link between the gizmo slider and each of the RGB channel sliders. Notice the three RGB channels turn bright green, indicating that a link is applied. Note that before I created these custom sliders, I first checked the range for the exposure slider nodes. Then, after creating the floating point slider, I entered the same range for the slider's min and max values. These range values can also be edited using the Edit button while creating a gizmo.

Once I had completed the gizmo, I exported and imported it back into a new script for testing, as seen in **Figure 18**. I composited it with an external flowchart that utilized several other passes, resulting in the image seen in **Figure 21**. This means the sss gizmo piped a subsurface effect into another flowchart, as well as provided all the necessary tools for adjusting its appearance using the knobs under the gizmo's user tab. Unfortunately, if you asked me what is it, I have no idea, only that I whipped it up for this article–good luck with any creative interpretation.

An additional feature that can prove handy when organizing knobs for a gizmo is the ability to add additional tabs to the gizmo. Essentially this means you can create sub tabs. From within the Manage User Knobs window, select Add… then choose Tab. Name the tab appropriately and notice how it is added as an additional tab to the gizmo. Any knobs below this tab will be included under that tab. An example for this would be separating embedded Read nodes from other knobs; hence, all the Read nodes *file* and *format* settings can be placed under a custom "files" tab, as seen in **Figure 22**, and then remaining knobs are placed under the user tab. Placing knobs under a tab (in the Manage User Knob list) forces those knobs to appear within that tab; hence, knobs above a tab will appear under the default user tab. Notice the default user tab is in fact the top tab and is added by default when you first create a knob; hence, it is located at the top of the list.

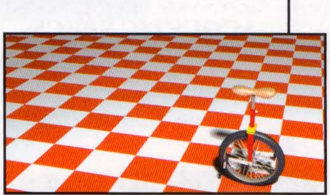
FINAL POINTS FOR CONSIDERATION

With respect to the previous article in this series (*HDRI 3D* Issue 6), this method can be used to simplify the creation process of an OpenEXR image that contains different passes embedded within separate channels. This entire conversion process can obviously be automated and hidden from the user. Once the gizmo is loaded, you simply locate the different passes that need to be appended, then by executing the gizmo, the OpenEXR image is compiled. The result: An OpenEXR image with all the passes embedded within, ready to take advantage of Nuke's strong abilities for multi-channel compositing.

A quick explanation for how this sort of gizmo can be assembled is as follows: First, create and label all the Read nodes appropriately (diffuse, reflection, specular, key light, etc.). The next step would be combining the Read nodes using Shuffle Copy nodes. Each time, you transfer RGB values from a Read node into a new layer that is named appropriately. For example, the specular pass Read node would connect through the A input into the Shuffle Copy node. In the Shuffle Copy node, create a new layer called specular with three channels (R,G,B) and transfer the specular RGB values into those channels respectively. Keep in mind that the B input is always the comp downstream and that the A input should always drive connections, meaning the B input travels downstream and Read nodes are applied using A inputs. If you are unfamiliar with this process, I recommend you review the first article in this series, Composting Passes with Nuke, *HDRI 3D* Issue 6.

Nuke really is a wonderful tool that provides easy control over some very difficult tasks. In this article we saw that we can easily create robust controls that can even be included as plugins under the Nuke menu.

Boaz Livny** has been working with 3D for over 10 years, on film, TV, and multimedia content. He is a technically savvy artist who specializes in modeling, lighting, and rendering, with experience working the entire pipeline. Boaz's studio,* ***www.visionanimations.com*, is located in NYC and provides regular services to clients and freelance support for studios. As a professor at NYU, he teaches Advanced Maya Courses and is currently involved in establishing an advanced training center in NYC–more details at* ***www.3darts.org****. He contributed to the next Sybex book,* Mastering Maya 7*, and is currently in the process of compiling his own mental ray book with Wiley.*



REVIEW: IBM IntelliStation M Pro

By Charles Edgin
Editorial Director, Teacher, Author
Henderson, Nevada

You have probably been asking yourself, should I get one of those new 64-bit workstations? To answer that question, you have to ask yourself how important your time is.

DO 64-BIT WORKSTATIONS REALLY DECREASE PRODUCTION TIME?

Let me start by giving you one answer that should answer all questions. Let's take rendering as one example of much time can be saved with a 64-bit dual-core workstation. In the performance test below using CINEBENCH 2003 v1, the render time for a Single CPU was 75.9 seconds and Dual CPU 40.8 seconds. That translates to 186% faster rendering. It is likely that in all areas of computer processing where dual-core can be utilized, this time savings should be there. Also, in running in 64-bit (compared to 32-bit), there is a 400% increase in processing, because a 64x64=>128 bit integer multiplication actually performs 4 times more work than a 32x32=>64 bit integer multiplication. Windows XP Professional x64 Edition initially supports up to 128 gigabytes (GB) of RAM and up to 16 terabytes of virtual memory. Physical memory support will grow in the future as hardware capabilities expand. Data in memory is accessed thousands of times faster than it is on a disk drive, offering a tremendous performance benefit for applications that are developed to use this greater system memory. These are all compelling benefits to move to a 64-bit workstation (**figure 1**).

The IBM IntelliStation M Pro tested was equipped with an Intel® Pentium® 4 at 3.2 GHz, dual-core processor, and Microsoft XP 64 Professional for the OS. The system came with 3 GB of ECC Memory and an Ultra320 SCSI Controller with a 10,000-rpm 73GB drive. It also came with an NVIDIA Quadro FX 1400; 128MB; 350 MHz core. I installed a SATA 40GB drive for purposes of testing (**figure 2**).

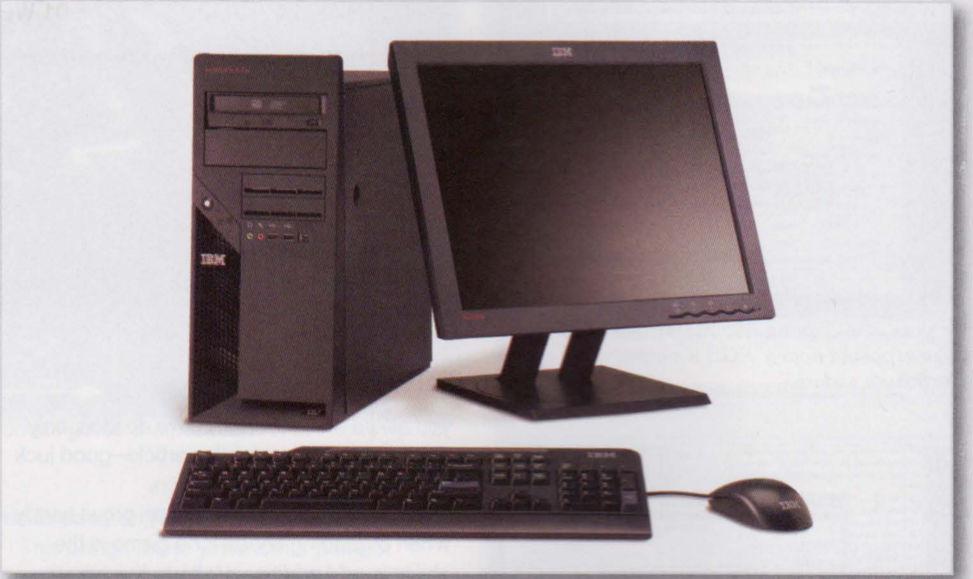


Figure 1.

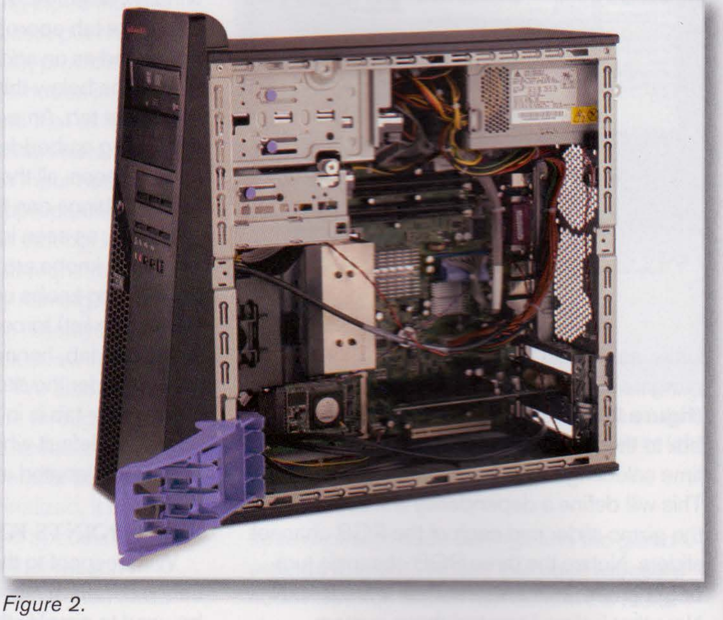


Figure 2.

PERFORMANCE AND IMPRESSIONS

My overall impression of performance is very good. Access information on the system is very fast. Access information through the Internet is very very fast because of Internet Explorer (64-bit). In all instances there have been no errors or failures.

The compact mini tower fits comfortably into snug workspaces, yet its flexible design packs up to a massive 1TB of data storage. Also, scale up with a maximum of 8GB of memory (PC-4200 ECC DDRII) with 800 FSB leaves you asking, what's not to like?

ACOUSTIC SPECS

This is the quietest system I have ever worked on. Only during boot-up is there any notable sound coming from the tower. Most of the time there is a light buzzing sound that is barely noticeable.

THERMAL SPECS

This is simple. At no time have I felt any heat coming from the tower. It always feels about 10 to 15 degrees cooler than my hand.

OS AND DRIVER ISSUES

As to be expected, with new computer technology come new systems—and new problems. Microsoft XP 64 is no different. The biggest problems are driver compatibility with existing printers, scanners, and so on. You need to weigh the value of the improved performance against possibly having to buy new peripherals or having to wait for new drivers to be released.

PROS

Reliability through IBM proven technology. In three months I have not had any crashes. With a background in testing and using computers since 1979, I have never had as reliable a system as this. The system has a clean and functional design. There are ample I/O ports, as shown in **Figure 3**. The system is very quiet and produces no additional heat to the office. When I touch the case, it is cool.

CONS

The only negative points are not with the M Pro, but with the lack of drivers running in Windows XP 64.

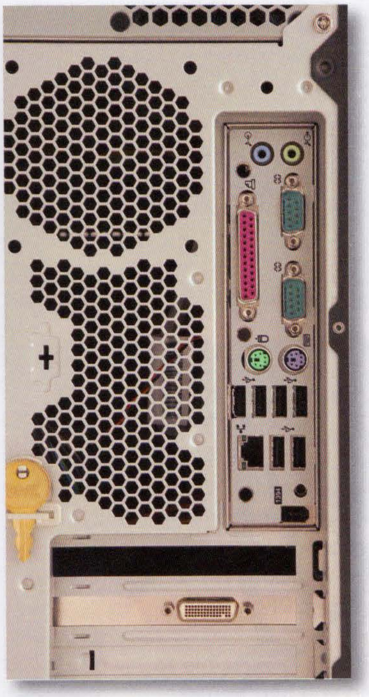


Figure 3.

SPECIFICATIONS (AS TESTED)

Processor: Intel® Pentium® 4 at 3.2 GHz, dual-core
L2 cache: 2MB L2 cache
Chip set/System Bus: Intel 955X Express, 800MHz FSB
ECC Memory: (std/max) 2, 8 1GB, 2GB/8GBDDR2 SDRAM/4 slots
OS: Microsoft Windows XP Professional x64 Edition (64-bit)
Advanced 3D: NVIDIA Quadro FX 1400; 128MB; 350 MHz core
Display: IBM model T117 TFT 17" monitor
Hard Drives: SATA: 40GB; Ultra320 SCSI: 73GB
Power Supply: 400W power supply

I/O PORTS

Network: Integrated Gigabit Ethernet with Wake on LAN®
USB 2.0: 8 (2 front panel, 6 back)
IEEE 1394: 2 (1 front panel, 1 back)
Serial: 2
Parallel: 1
Audio: AC-97, mic in (front and back) line in/out, integrated speaker
4 x 4 Minitower: (1) PCI Express x16
4 Slots/3 Open: (1) PCI Express x4 (x8 mechanical) (1) PCI Express x1 (1) PCI 32-bit/33MHz
4 bays/2 open: (1) 3.5" internal (1) 3.5" external (optional floppy drive) (2) 5.25" external

The IBM M Pro, as configured, is priced about $3,387 with a 17-inch monitor.

ACOUSTIC SPECIFICATIONS

Idle - 4.3 bels or about 43 dB (about the sound level of a "residential area at night")

70% CPU load - 4.8 bels or about 48 dB (about the sound level of a "quiet restaurant")

THERMAL SPECS (SMITHFIELD)

a. TDP:30W, Performance FMB(2005)
Thermal spec is: Tc = [P*0.26]+42.1 = 75.9° C or 168.6° F
b. TDP:95W, Mainstream FMB
Thermal spec is: Tc = [P*0.28]+44 = 70.6° C or 160° F
c. TDP:70W, Mainstream FMB
Thermal spec is: Tc = [P*0.28]+44 = 61.22° C or 142° F

Thermal Design Power (TDP)
Flexible Mother Board (FMB)

TESTING:

SPEC Viewperf 8.1

3dsmax-03 Weighted Geometric Mean = 31.28
catia-01 Weighted Geometric Mean = 26.91
ensight-01 Weighted Geometric Mean = 18.66
light-07 Weighted Geometric Mean = 22.65
maya-01 Weighted Geometric Mean = 46.20
proe-03 Weighted Geometric Mean = 48.21
sw-01 Weighted Geometric Mean = 25.50
ugs-04 Weighted Geometric Mean = 29.26

CINEBENCH 2003 v1

Processor: Intel® Pentium® 4 at 3.2 GHz, dual-core Intel 955X Express, 800MHz FSB
MHz: 3.2
Number of CPUs: 2
Operating System: Microsoft XP 64 Professional

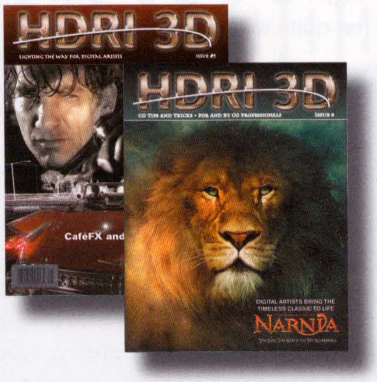
Graphics Card: NVIDIA FX 1400
Resolution: 96 DPI 1280 X 1024

Render Speed:
Rendering (Single CPU): 347 CB-CPU 75.9 sec
Rendering (Multiple CPU): 645 CB-CPU 40.8 sec

Multiprocessor Speedup: 1.86

Shading (CINEMA 4D): 396 CB-GFX
Scene 1: 3.39 sec; 30.1 fps *Scene 2:* 1.13 sec; 79.1 fps
644565 polygons/sec
Shading (OpenGL Software Lighting): 1553 CB-GFX
Scene 1: 6.88 sec; 14.8 fps *Scene 2:* 2.90 sec; 31.1 fps
2117857 polygons/sec
Shading (OpenGL Hardware Lighting): 3495 CB-GFX
Scene 1: 3.39 sec; 30.1 fps *Scene 2:* 1.13 sec; 79.1 fps
5426658 polygons/sec

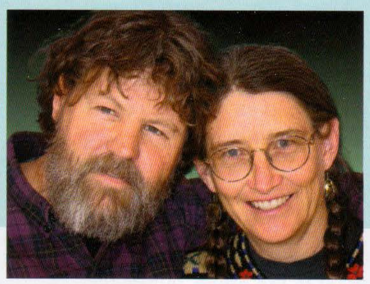
OpenGL Speedup: 8.82



HDRI 3D back issues
$7.95 each
when you subscribe online at
www.dmgpublishing.com
(Except issues that are almost sold out.)

FINAL RENDER:

WHAT'S IN A NAME?



By Andrea Carvey
Archeologist, Animator
New Mexico

Let's say that you've just created a model or animation and now need to save it. What do you name the file? Well, the most obvious choice is some sort of descriptive name, like "Box." Pretty straightforward. But then, what if you create another model and/or animation of a box that is different from "Box" in some way? To avoid replacing "Box" with the new one, you have to save it with a different name. How about "Box2"? So far so good.

Now, what about creating different versions of "Box2"–say one that is blue, one that is green, and one that is black? Well, that's easy. Just name them "Box2_Blue," "Box2_Green," and "Box2_Black." Simple! That's what I do...keep it simple.

That's all very well and good, but consider the possibility, nay probability, that your client likes "Box2_Green," but wants to make a few little changes.

"Can you make it a little more shiny?" he asks.

"Oh, sure," you say confidently.

So, you increase the specularity and glossiness and save it as "Box2_Green_Shiny."

"How about not quite so hard-looking?"

"OK."

And you tone down the glossiness a bit ... "Box2_Green_Shiny_02."

"That's looking really good, but could you add some bumps now?"

"I'll try."

"Box2_Green_Shiny_02_Bumps."

After a few long-term projects, it occurs to you that it might be beneficial to add the date to the name, hence "Box2_Green_Shiny_02_Bumps_20Feb." That's what I do. (If you don't know why, don't worry, you will find out).

As you can imagine, things can start to get a little unwieldy. But you persevere.

Apparently, you have nothing better to do, so you decide to tweak the box a little bit. Maybe you want to try out a plugin that you heard was particularly sweet. But how do you name that? It is just a test, but it is convenient to preserve the "genealogy," so to speak; therefore you come up with "Box2_Green_Shiny_02_Bumps_20Feb_TestA."

Eventually, you have several iterations of your test, and a bunch of variations of those iterations, one of which you really like. Now what do you do? "Box2_Green_Shiny_02_Bumps_20Feb_TestAc_ver2.5."

In the olden days, it was impossible to have long filenames, so you had to use abbreviations, which could get rather cryptic, like "Bx2GS2B." You may even be tempted to use such a method in this case; however, be forewarned, in a few months the present project will be but a shadowy memory. And one day, as you are just settling down in the hammock on a nice, warm, spring day with a cool drink in your hand and a nice BLT, you get a call from the client requesting a few more adjustments. No problem, you think.

However, if you can even find the hard drive where you put all that stuff, you are doing well! Then you come across "Bx2GS2B."

"What the...?"

It seems vaguely, yet terrifyingly, familiar. Like the feeling you get when someone calls you by your full, even-your-middle-name-that-no-one-but-your-immediate-family-knows, name. Not that the name "Box2_Green_Shiny_02_Bumps_ 20Feb_TestAc_ver2.5" clarifies anything. Especially in light of the fact that there are also versions 2.6, 2.7a, 2.7b, and 3.0.

"Hmm... now, which one did I send the client?"

And for even more fun and enjoyment, suppose that you broke all rules of logic and sanity and didn't name the rendered animation with the same bulky name as the scene that rendered it!

"Let's see...do you suppose the animation I want is "Box_Final_2.5," or maybe "Box_Final_FINAL," and did I use "Box2_Green_Shiny_02_Bumps_20Feb_TestAc_ver2.5" to render either of them?

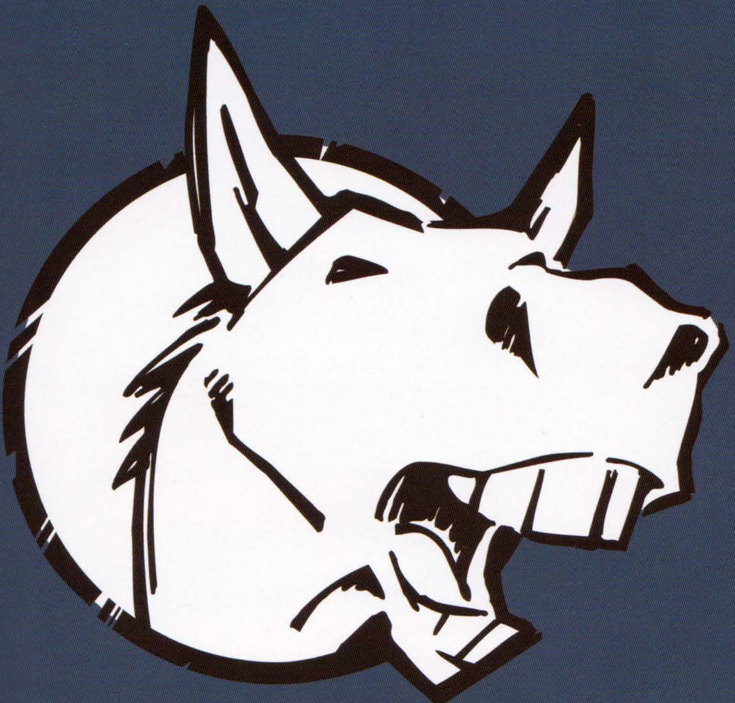
If you have done what you should have done, you will be able to refer to your tidy notes, where you have recorded every excruciating, mind-numbing, detail of your process for that project...of course. (You know, the same kind of notes that are taken during an autopsy...at least, how it's portrayed on TV.) Except that during the last few weeks of the project, things were so hectic that you neglected to enter anything into your notes, fully intending to spend a few hours after final delivery to diligently record everything. But it never happened.

So, alas, you are stuck trying to reconstruct the dreaded Scene/Animation hierarchy, only now, because you didn't spend those few precious hours when you had them, it will take days, which you don't have.

At least, that's what I do.

***Brad and Andrea Carvey** have been doing computer animations for a long time. In 1969, Brad used an analog computer, which was the size of a car, to produce his first computer animation. Andrea, an archeologist, prefers to do scientific animations. Her credits include programs like The Discovery Channel's* Understanding Cars. *Brad is an electrical engineer and an Emmy Award-winning member of the Video Toaster development team. He prefers to do feature film work. His credits include films like* Men in Black, Stuart Little, Black Hawk Down, Kate & Leopold, *and* Master of Disguise.

mule 2.0



Download your FREE
unlimited 30 day trial today!

www.epicsoft.net/HDRI/mule

LIGHTWAVE . NETWORK . RENDERING . THAT . HAULS . ASS

epicsoft™
Advancements For The Business of Creativity

Lightwave and LightWave 3D are trademarks of NewTek, epicsoft and MULE are trademarks of epicsoft LLC.

interaction
face 2B0ST0N6

Mk Haley | Bostonian | BFA University of Massachusetts | MFA Cal State Los Angeles | Technical Advisor, Walt Disney Imagineering Creative Development, Glendale, California | 18-year SIGGRAPH attendee

Masa Inakage | MFA California College of the Arts | Professor, Keio University Media Design Program, Kanagawa, Japan | 22-year SIGGRAPH attendee

5-days of real-world, real-time graphic, interactive twingularity

The only conference and exhibition in the world that twingles everybody in computer graphics and interactive techniques for one deeply intriguing and seriously rewarding week. In Boston, where thousands of interdisciplinary superstars find the products and concepts they need to create opportunities and solve problems. Interact with www.siggraph.org/s2006 to discover a selection of registration options that deliver a very attractive return on investment.

SIGGRAPH2006

The 33rd International Conference and Exhibition on Computer Graphics and Interactive Techniques

Conference 30 July - 3 August 2006 Exhibition 1 - 3 August 2006 Boston Convention & Exhibition Center Boston, Massachusetts USA

IMAGE CREDITS: limosa © 2005 Brian Evans; moo-pong © 2005 Jun Usu, Daisuke Uriu, Naohito Okude, Keio University Okude Laboratory; 2005.1 © 2005 Kenneth A. Huff

Sponsored by ACM**SIGGRAPH**